

CENTRAL CIRCULATION BOOKSTACKS

The person charging this material is responsible for its renewal or its return to the library from which it was borrowed on or before the **Latest Date** stamped below. **You may be charged a minimum fee of \$75.00 for each lost book.**

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

TO RENEW CALL TELEPHONE CENTER, 333-8400

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

DEC 14 1998

JUL 05 2000

When renewing by phone, write new due date below previous due date.

L162



Digitized by the Internet Archive
in 2013

<http://archive.org/details/pdp11multiplexor942whal>

570.57
IL6r
no. 942

UIUCDCS-R-78-942

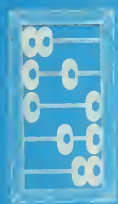
UILU-ENG 78 1735
C00-2383-0054

A PDP11 MULTIPLEXOR

by

A. D. Whaley

September 1978



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-78-942

A PDP11 MULTIPLEXOR

by

A. D. Whaley

September 1978

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

Supported in part by the U. S. Department of Energy under grant US ENERGY/
EY-76-S-02-2383.

Acknowledgements

Several people contributed in various ways to the development of this device. Early ideas including principally the storing of device registers in semiconductor memory came from Greg Chesson, and grew out of his discussions on the same subject with the late D. B. Gillies. The idea of using a priority matrix instead of a linear scheme is due to Lawrence Lopez. Implementation of my sometimes less than specific architecture and careful working out of the failsafe system was done by Bruce Artwick who also deserves credit for making sure the prototype actually got constructed.

Table of Contents

1	Introduction	3
2	Functions and Capabilities	4
3	Physical Structure, Functional Components	6
4	Multiplexor States and Functions	8
4.1	Idle loop, S0 and S0A	8
4.2	I/O state, S1	9
4.3	Delay for active, S102	10
4.4	Active error, S101	10
4.5	Read state, S3	11
4.6	Save input character, S12	12
4.7	Transfer word of input data, S11	13
4.8	Transfer single byte, S13	13
4.9	Increment address, length, S24	14
4.10	Check for transfer complete, S15	16
4.11	Complete transfer at device, S16	17
4.12	Store device registers, S5	17
4.13	Wait for device to reset, S5A	18
4.14	Write state, S4	18
4.15	Transmit saved byte, S13	19
4.16	Fetch from memory, S19	19
4.17	Register operation, S2	20
4.18	Load register from CPU, S2F	21
4.19	Register write, wait for completion, S2A	21
4.20	Register write, completion detected, S2C	22
4.21	Register read and wait for completion, S2B	22
4.22	Register read, completion detected, S2D	23
4.23	Transfer leftover byte, S2E	23
5	Data Registers	25
5.1	Data paths, RAMS	25
5.2	Slow access to by computer	25
5.3	UTRAP, shared resources concept	26
5.4	Addressing	27
6	Device Control Signals	32
6.1	Set request	32
6.2	Echo	32
6.3	Set active	33
6.4	Reset	33
6.5	Zap	33
6.6	Quit	33
6.7	Final	33
7	Protocol at Devices	34
8	Priority Select Matrix	36
8.1	Design choices	36
8.2	Matrix structure	36
9	Software Compromise	39
9.1	Definition of ASCII Control Characters	39
9.2	Delayed data store	39
9.3	Delay before echo	39

10	Data Transfer on UNIBUS	41
11	Interrupts	43
12	Error Lights	44
13	Automatic Echo	45
14	Proposals	46
15	Kinds of Interfaces	48
15.1	Speed Select LIA	48
15.2	Busy Interlock	48
15.3	Telephone Dialer	48
15.4	Clock	49
15.5	Paper Tape Readers	49
15.6	FIFO Card	49
16	Conclusions	50
Appendix A	-- cost breakdown	51
Appendix B	-- failsafe system	55
Appendix C	-- address plugs	57
Appendix D	-- EIA interface diagrams	60
Appendix E	-- Tables	66
Figure 1.	Physical layout	7
Figure 2.	Data flow of transmitted character	28
Figure 3.	Data flow of first device register	29
Figure 4.	Data flow of second device register	30
Figure 5.	Data flow of received character	31
Figure 6.	Timing of main control signals to device.	35

1 Introduction

In recent years, inexpensive hardware has been proliferating. At the same time more complicated hardware has been proliferating as well, as complex hardware may now be built more cheaply, compactly, and reliably. Multiplexors are now becoming quite common on minicomputers, but they are almost exclusively specialized for serial applications. This report describes a multiplexor that offers a slightly higher per terminal add-on cost than serial multiplexors, though still inexpensive, and which provides almost complete freedom in the type of equipment that may be supported.

This multiplexor in essence is a bus converter. The philosophy used in design was to have the multiplexor provide a wealth of difficult functions, converting a very general minicomputer bus into an extremely simple bus to which greatly simplified peripheral interfaces could be attached. Devices do transactions of eight bit bytes of data, one at a time, and are told when to be quiet. They are given a few option signals (which they must store) to control their behavior. All the difficult tasks are performed in the main multiplexor unit by one set of circuitry which serves all. This circuitry exists once, rather than being duplicated for each of the 256 devices.

Although features are included to facilitate handling of terminals with human typists seated in front of them, the design of the multiplexor is quite general, and its characteristics quite versatile.

This report does not discuss the circuitry of the multiplexor extensively, instead giving an overview of its nature and some of the rationale for design.

2 Functions and Capabilities

As designed, the multiplexor was to have several characteristics, defined primarily in terms of the external properties it should exhibit. Many of the properties may be seen from the nature of the device registers.

It was decided that the computer should be presented with separate device registers for each device. With two words - four bytes - for each device, and 256 devices, a total of one thousand bytes of address space are used. The format of a device register for each device is:

write:	+2	ctrp	opl	op2	actv	adr 17, 16	length 9 - 0
	+0	address 15 - 0					
read:	+2	ctrp	opl	op2	actv	adr 17, 16	length 9 - 0
	+0	** undefined **				sense 1	sense 2

Of the obvious features which may be seen from the registers are a full eighteen bit memory address, and a single transfer length of up to 1023 bytes (actually 1024 if the transfer is initiated with length already set to zero). With the address and length maintained separately for each device, up to 256 transfers direct to memory are allowed to operate concurrently.

ACTV is the active bit; setting it from the CPU causes a transfer to start, and the transfer is forced to stop if the bit is cleared from the CPU at any time. The bit is cleared by the multiplexor automatically if the length register becomes zero when decremented after transfer of a byte to or from a device. The bit is also cleared if CTRAP (control trap) is set, if the transfer in question is a read operation, and if the character received is an ASCII control character. CTRAP does not affect output transfers.

Direct communication to the device is provided in both directions. OPl and OP2 are set by the computer and interpreted individually as the device sees fit. The computer may read bits SENS1 and SENS2. As each interface has both an even-numbered device for input and an odd-numbered device for output, four options and four sense bits are available for any purpose within each interface.

Internally, nine bits are saved per device, eight to buffer a single character and one to remember whether a character is being saved. These nine bits with the thirty two in the two device registers

described above, bring the total of bits saved for each device to forty one. For 256 devices this represents a formidable amount of information; it is kept in semiconductor memory.

In addition to allowing access to device registers, and transferring data, the multiplexor contains an interrupt facility. When the multiplexor chooses to reset ACTV on its own initiative (at the end of a transfer), the device number of the device in question is entered into a queue. The queue then independently attempts to cause an interrupt. When a special multiplexor device register (not one of those previously mentioned) is examined, the computer is given the oldest device number in the queue and that entry is discarded. Clearly the queue must not be examined capriciously. Since an interrupt does not identify the causing device, the computer must examine this special register to determine which device completed a transfer. The queue continues to cause interrupts until it is empty.

The interrupt queue contains sixty four positions. Clearly, if sixty five transfers complete simultaneously, there is every chance that the last will be forgotten because the queue will overflow. Not only is this extremely unlikely in practice (it has never been observed to occur), but an error condition is preserved for this event so it would not go undetected. The size of the queue can be easily expanded, if necessary.

A special echo control signal is provided by the multiplexor during input transfers to help devices echo input automatically at their option. The signal is not generated for control character codes.

One last feature provided is that of initialization. When the computer sets the INIT line, not only does the multiplexor properly reset itself, but it also enters a loop of selecting each device in turn and resetting it individually.

3 Physical Structure, Functional Components

The multiplexor comes in several packages. The main unit interfaces to the minicomputer bus and is contained in the usual package used by the manufacturer for the purpose (a "system unit"). A cable identical to that used for the minicomputer bus goes from the main unit to a distribution rack, a standard nineteen inch by 5 1/4 inch rack with four printed circuit cards and room for nine bus cables to device racks. Each device rack resembles the distribution rack and may contain fifteen device cards except the ninth which contains only eight (see figure 1). Standard components from the manufacturer have been used throughout for printed circuit card racks, bus cables, and the system unit. The printed circuit cards are from a large independent supplier. The wirewrapping is done by a large reliable company, Datacon Inc., of Woburn, Massachusetts.

The electrical layout is similar to the physical layout. The multiplexor handles transactions on the minicomputer bus and communicates with the devices via a much simplified "device bus". The distribution rack drives the signals to the many device racks at a sufficiently higher power level and also converts an eight bit port number into individual select lines for each of the 256 devices. The modules have all been designed so that they may be unplugged from each other without causing damage to components. The device racks and interface cards may be unplugged without installing terminators and the unit will still operate correctly.

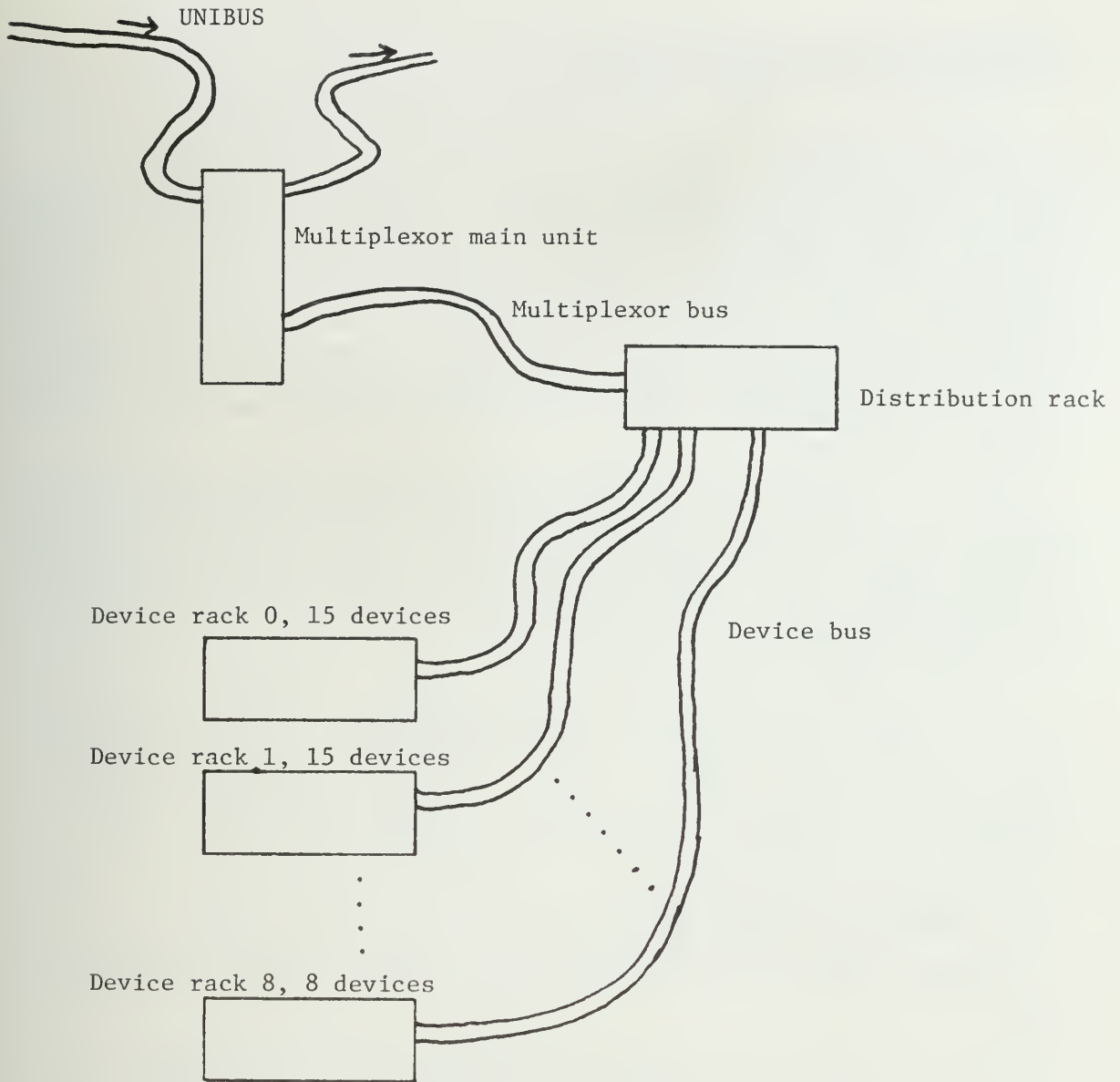


Figure 1

4 Multiplexor States and Functions

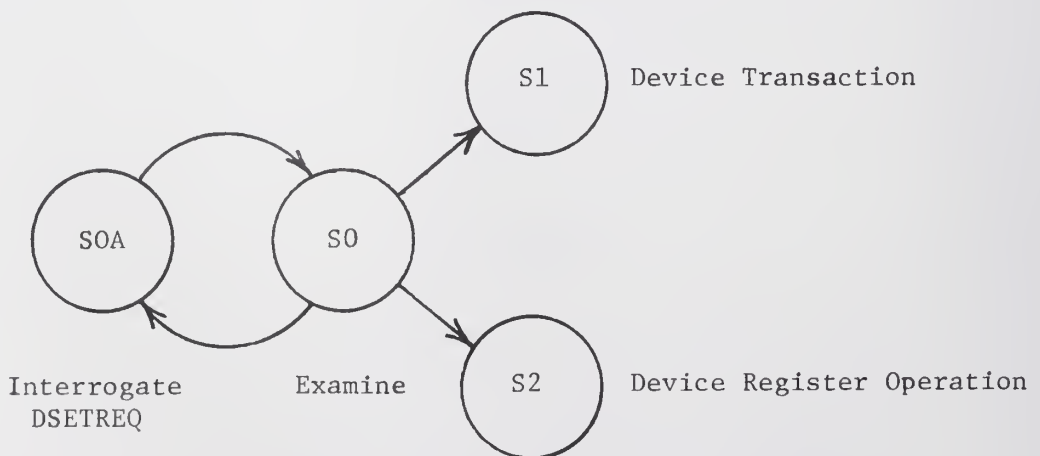
The multiplexor control circuitry may be said to resemble a program or a state diagram. Each state or program step performs a specific function. A state has associated with it a state flipflop, a state time during which the state flipflop is turned on (and all others are turned off) and a state signal, the output of the logical one side of the flipflop. The state signal is therefore a short pulse for the duration of the state, useful for activating whatever circuitry is supposed to be used during the state. One of the uses of each state signal is to activate the next state one state time hence. The states constitute steps in a wired program. Usually the program is executed synchronously for simplicity, each state having the same duration, but there are important exceptions.

One side effect of the program-like structure is a fairly long processing time for multiplexor functions. This can be an advantage, since the processing time is still much faster than needed, but prevents the multiplexor from creating memory interference more than every few microseconds. This fact can help prevent overruns on faster devices such as disks and drums.

Interesting characteristics of the program are an interrupt facility called UTRAP (more closely resembling abandonment than an interruption), and a devices lockout for read-modify-write computer interrogations called DATIP. These peculiarities will be discussed elsewhere in this report. This chapter is concerned with the functions of the states themselves.

4.1 Idle loop, S0 and SOA

The two states S0 and SOA form a state machine loop waiting for something to do:



There are two kinds of requests that could occur. One kind is that given by a device to perform some transaction (read, write, terminate).

The other is that given by the central processor to operate on the device registers. These two operations must be mutually exclusive as the device registers are stored in a single port memory; the port may not be shared by two functions simultaneously.

State SOA is used as an interrogation signal to determine whether any requests are outstanding. A one-bit register (called USELF) is set to one if the CPU is attempting to access a device register. In addition, the interrogation signal is sent to all devices under the name DSETREQ. Each device has a one-bit register it loads on this command with its requesting/not requesting status. Each device uses the output of this register to allow itself to enter its request into the priority selection matrix. Thus, no device will make a request (or stop making an old request) except synchronously on receipt of the signal from SOA.

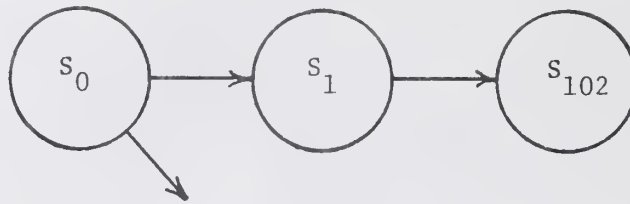
State S0 does nothing whatever. This allows time for the signal from SOA to reach all devices, and to permit the priority selection matrix to settle on a choice among the requesting devices. The major delay is caused by the round trip cable transmission delays to and from the main multiplexor unit. At the end of state S0, some simple gating returns control to SOA to repeat the interrogation process unless a request has been found. In that event, control proceeds instead to S1 or S2 as appropriate. State S2 has priority by necessity since it implies that the CPU has control of the computer bus, thus preventing the multiplexor from transferring data.

4.2 I/O state, S1

After S0 determines that a device request is being made, control proceeds to S1. The main purpose of S1 is to load a set of easily accessible device registers from the RAMs (solid state memory) where they are stored. By the time state S1 begins, the exact number of the device making the request must be available from the priority selection matrix.

A register is loaded with the device number in the priority select system during S1. This prevents the number from changing while the request is being honored, an event which would cause chaos (for example, if the device register contents for one device were written back later into the RAMs in the position for some other device).

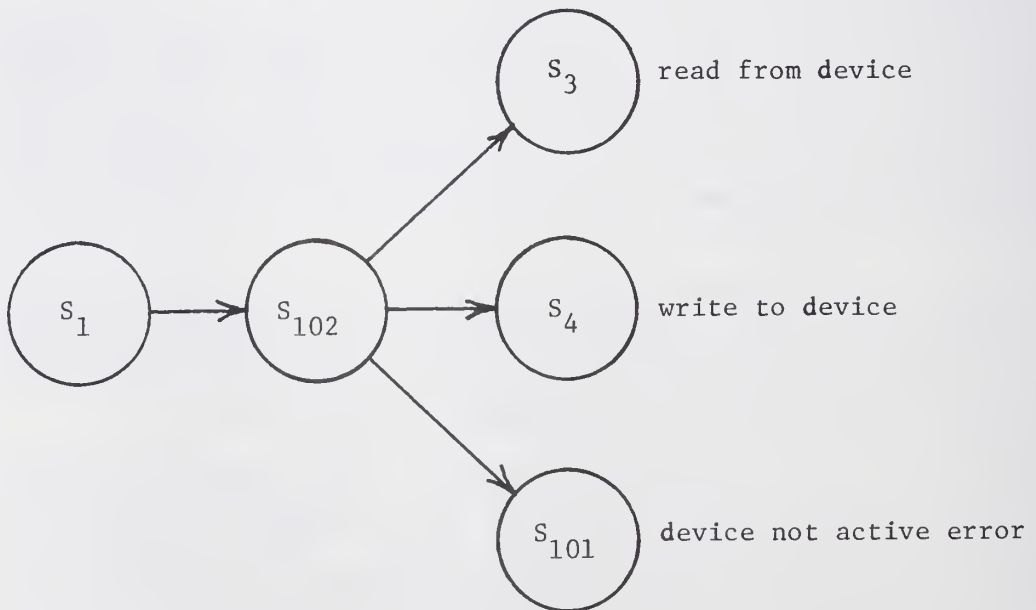
The output of the priority select matrix in fact could not change anyway if all devices behaved correctly, i.e. altering their requesting status only on receipt of the set request signal (SOA). Devices are considered to be of much lesser integrity, however, due to their large numbers and sometimes experimental nature. It is in any case considered good design practice to have a self contained module assume a healthy skepticism for the "outside world." Devices are also to be constructed at a later date by the users of the multiplexor, placing them in the "customer domain." The multiplexor should, therefore, protect itself against particularly hard to analyze errors.



4.3 Delay for active, S102

State S1 loads a set of device registers with the appropriate contents from a fast memory. The state time is considered to be reserved for memory access delay, and the device registers are loaded at the end of the state. Since the next state to be executed, either S3, S4, or S101, depends in part on a bit in the device register (the active bit), a race condition exists testing a bit just as it is being made available. State S102 was inserted merely to avoid this condition, by marking time for one clock cycle.

State S102 also does a sample and hold on signals DQUIT and DFINAL for use in later states.

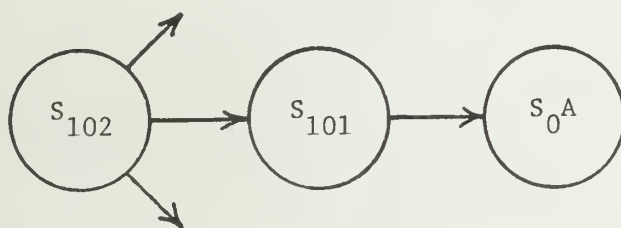


4.4 Active error, S101

One of the more difficult errors to detect would be that of a byte being transferred to an inactive device. This would cause data on a read device to be placed in the computer memory at a completely erroneous location, using an address that merely happened to reside in the address register. When a supposedly inactive device makes a request S101 is

entered, which retransmits the correct contents of the active bit and option bits to the device in an attempt to convince the device to cease. An error light is also turned on (one of two lights), and continues to be on until the computer is restarted (INIT condition).

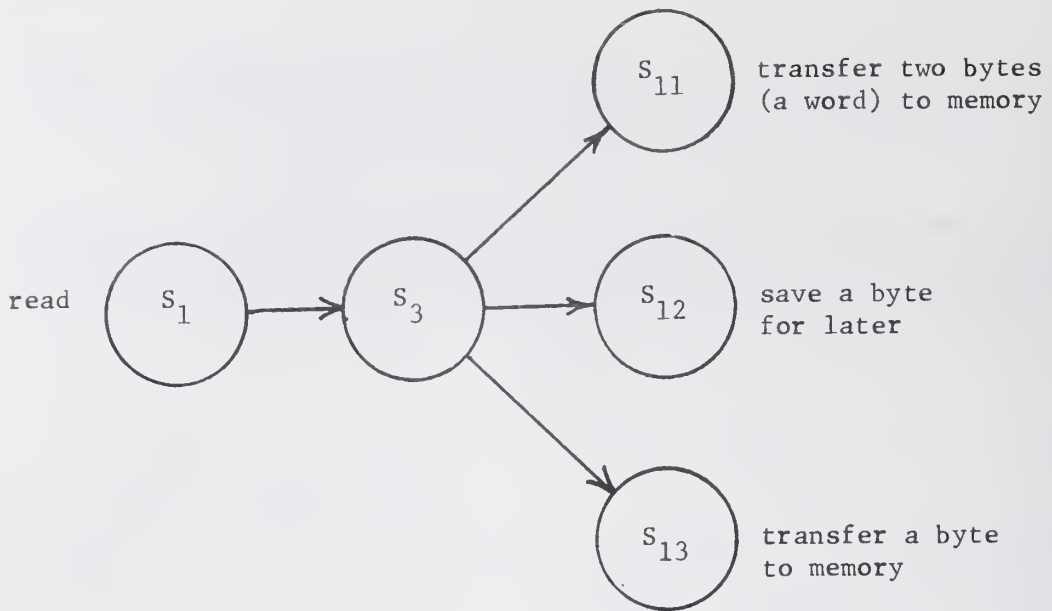
Quite a few kinds of errors in a device or in the priority selection matrix can appear to the main multiplexor unit as this kind of error in the active bit. The ease with which this error can occur and the potentially disastrous nature of the results make this simple test a must.



4.5 Read state, S3

This state is present merely to provide additional conditional branching to three other states. Before S102 was added, S3 and S4 also provided a useful delay function, but they could probably now be eliminated, with state S1 performing a larger multi-way conditional branch.

The conditional branching depends completely on buffering considerations. When possible, the multiplexor accumulates pairs of bytes to be sent to the computer memory as words, thus halving the memory contention. This is possible everywhere except potentially on the first or last byte of a block of data. If, when state S3 is entered, the data being transferred is just beginning a word boundary and more data will follow (transfer length greater than one), then the byte from the device is saved in the device register (state S12). If the byte is to be stored in memory by itself, state S13 is entered for this purpose. If a byte was previously saved so that with the currently available byte a complete word is to be stored in memory, state S11 is entered.



State	Conditions for Entering	Actions
S11	Save bit on	transfer a word, clear save bit
S12	save bit off; transfer not terminating; on word address boundary	save byte; set save bit
S13	save bit off; state S12 not being entered	transfer byte

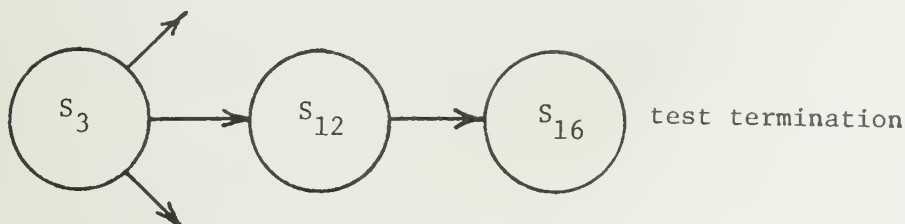
4.6 Save input character, S12

As mentioned above, S12 is entered if it is appropriate to save the byte just made available in the device register for transfer later. Loading the "save" part of the device register automatically also sets the ninth (save) bit in the device register indicating that a byte has been saved.

State S12 also sets a flipflop called MUX3CTRL which controls the source from which data is loaded into the save portion of the device register (in this case from the device character bus). The flipflop is not reset until S5 in order to avoid a race condition at the precise end of state S12 at which time the actual device register loading is performed. The loading must be performed at the end of the state in order to allow the multiplexor controlled by MUX3CTRL to establish itself.

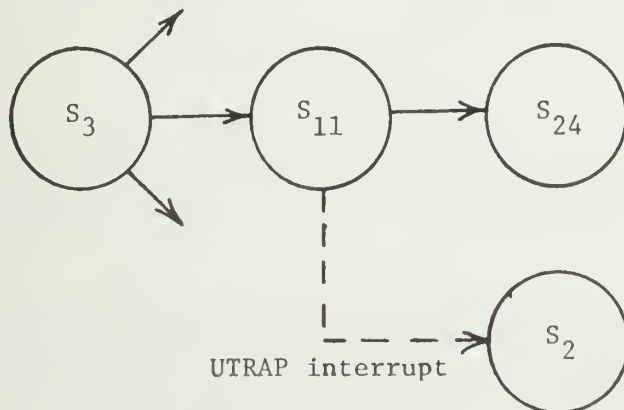
If DQUIT is not present, DECHO and the address increment signal are

generated as discussed for state S24.



4.7 Transfer word of input data, S11

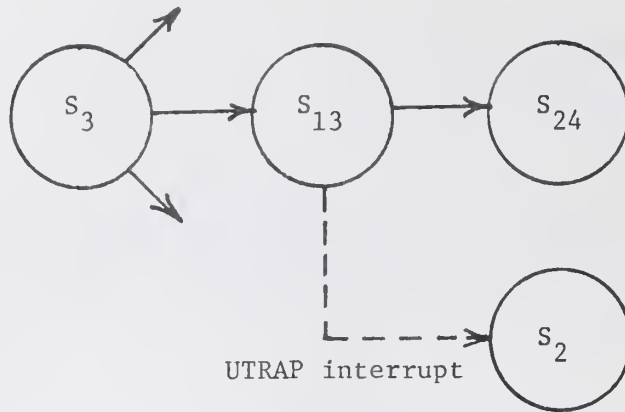
As discussed, state S11 usually transfers a current and a previous byte of data into a word of computer memory. If signal DQUIT is present only the previous SAVE byte is transferred. As one of three states doing data transfers onto the computer bus, its timing is complex (see the section on bus data transfer). S11 also resets the save bit indicating data is no longer being saved. This might be considered to be an error since execution can be suspended at S11 and restarted from SOA later (see UTRAP). However, when execution is suspended, the contents of the device registers are discarded and later reloaded afresh from solid state memory, so resetting of the save bit would also be discarded.



4.8 Transfer single byte, S13

The single byte transfer, S13, differs from state S11 in three main ways. It causes the bus signal C0 to be set during data transfer, telling memory only one byte is being transferred. It also enables sending the lowest (even/odd) address bit to the address portion of the bus. This bit is forced to zero otherwise. It further guarantees that the byte to be transferred is available in both halves of the data word sent to the bus. This follows the PDP11 bus convention that during a byte store, the data must still be in the correct halfword position depending on the even/odd address bit.

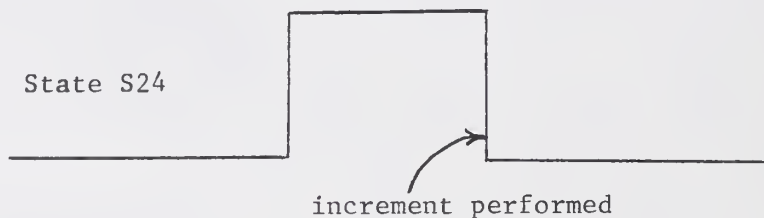
If signal DQUIT is present, no transfer takes place at all.



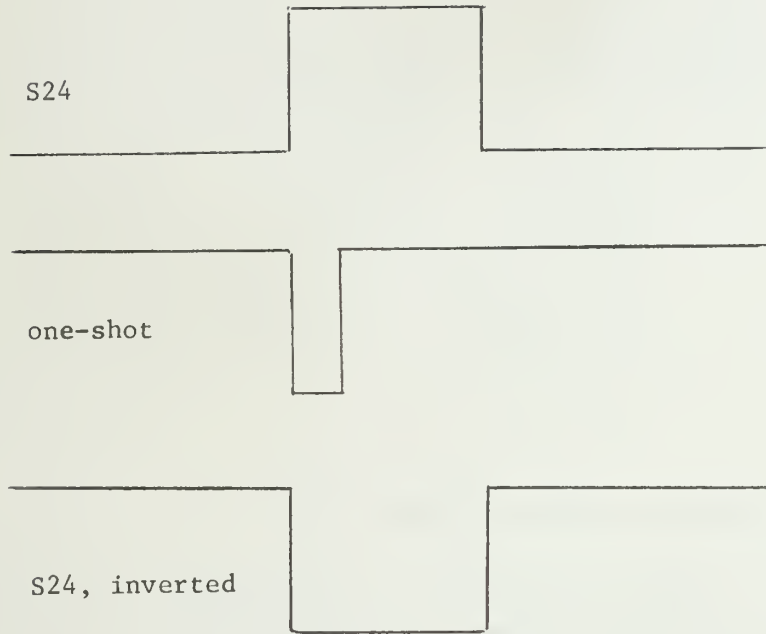
4.9 Increment address, length, S24

State S24 increments the address and decrements the length after a data transfer. The increment must be performed after the data transfer so that the proper address is used in the transfer; states S11, S13, and S19 cannot therefore perform the increment themselves. This same state signal is also used to form DECHO, if appropriate, and is suppressed for both functions if DQUIT is present.

The signal to the address and length counters causes an increment on the positive going edge, so the state signal for S24 itself was originally used for increment.

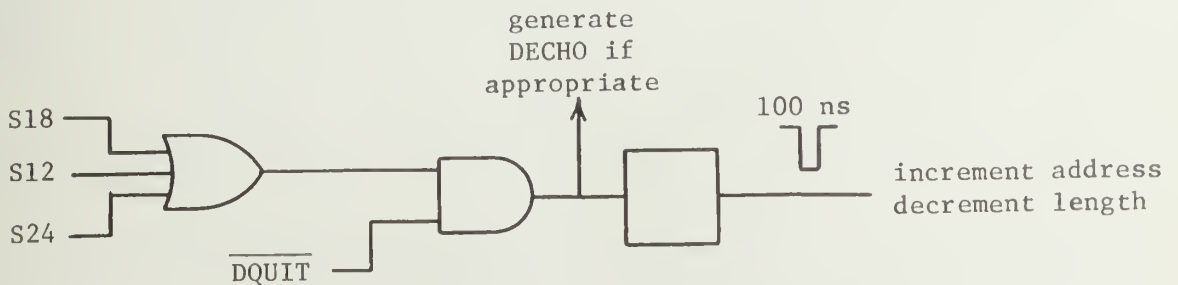


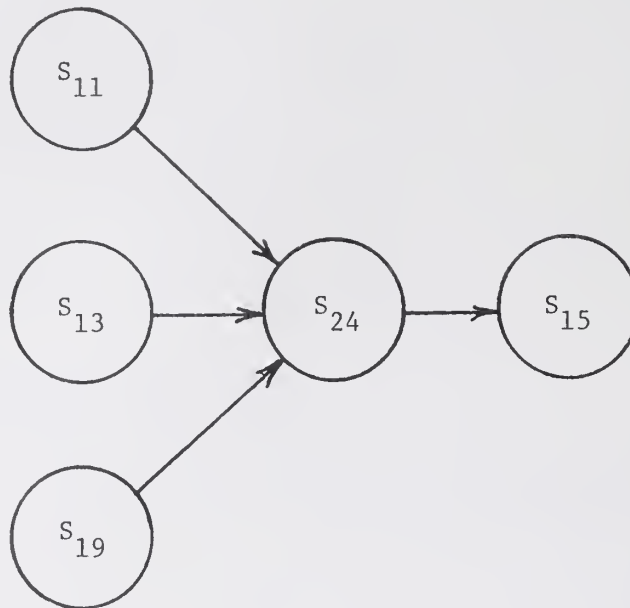
Because of the design of the integrated circuit used for the counter however, loading part of the counter caused spurious data to appear in the other part. Following the manufacturer's recommendations led to the employment of a one-shot to generate a pulse:



The inverted S24 signal has the right shape but cannot be used since the increment must be performed well before the end of state S24, not at the end. This is to avoid a race condition as the next state tests the incremented result.

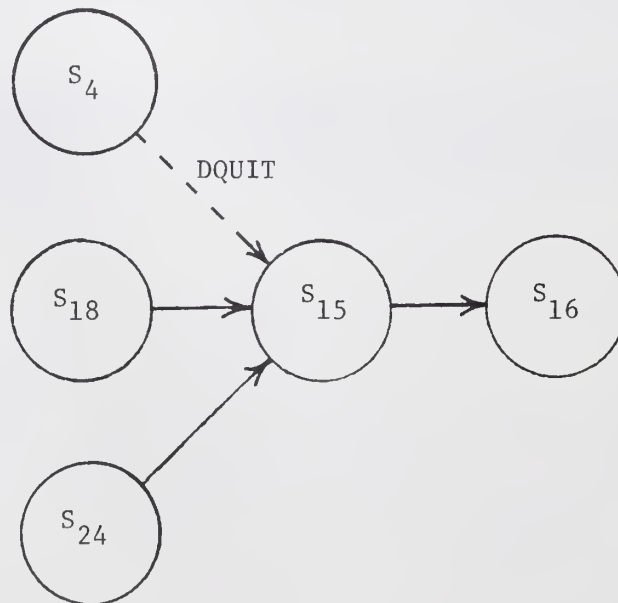
The duration of the one-shot may be shortened to barely what is required by the particular counter (74193) used, and will not depend on any other timing. Currently a lengthy 100 nanoseconds is being used.





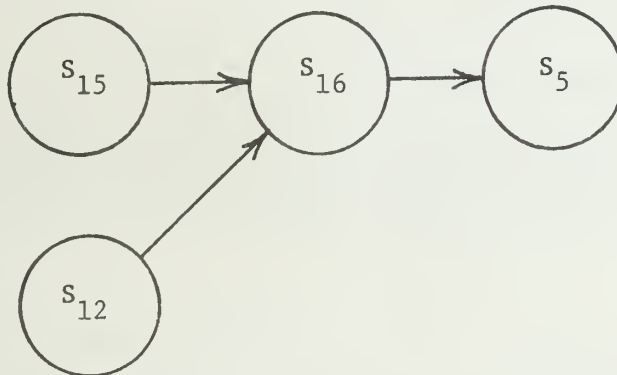
4.10 Check for transfer complete, S15

End of transfer is considered to be a remaining transfer length of zero, or a control character on a read operation with CTRAP enabled in the device register or either of signals DQUIT or DFINAL is present. These conditions are gated with S15 to generate two signals, one to reset the active bit and one to cause an interrupt indirectly by pushing the current device number into the interrupt queue (FIFO).



4.11 Complete transfer at device, S16

This state generates the signal DRESET, which is sent to all devices on the device bus. Supposedly, only the device currently selected by the device bus is paying any attention to DRESET. If the device is a read device, it causes the device to abandon the character it is sending to the multiplexor on the device bus data lines and to prepare for another character. Since the previous state, S15, is testing the character, the potential removal of the character in S16 is the basic obstacle preventing the two states from being combined. DRESET is not issued if DQUIT is present.



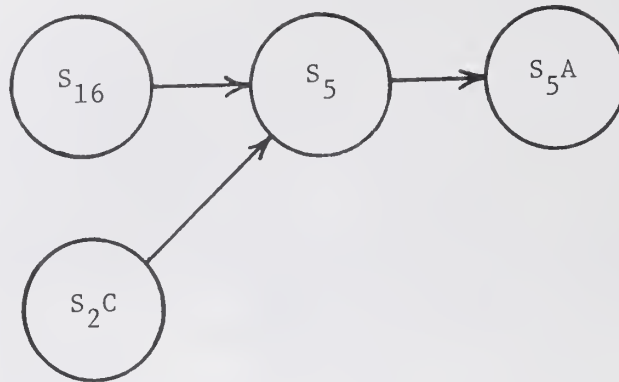
4.12 Store device registers, S5

After a transfer has been completed, various parts of the device registers have been changed or updated. State S5 writes those registers back into the solid state memory in the position reserved for this device. The device number was originally loaded in state S1 or S2 into a register located in the priority selection hardware. This is important if execution comes from state S2, as the device number would definitely not still be available otherwise. From state S1 the device number would still be available barring errors in the devices.

Another function of S5 is to generate the device bus signal DSETA (or "set active") which is intended to inform the selected device to load in a new fresh contents of its active bit and two option bits. If the transfer has just finished, the new contents of the active bit will be zero. The option bits will be identical with the previous bits stored in the device if S5 is entered from S1, but may or may not be changed if from S2.

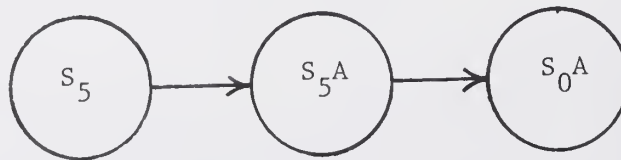
Note that execution from S2D does not go through S5 as S2D is used only to display device registers to the central processor, and no modification takes place.

State S5 also resets the save bit in the device registers if the active bit is no longer present; this is done only as a safety measure.



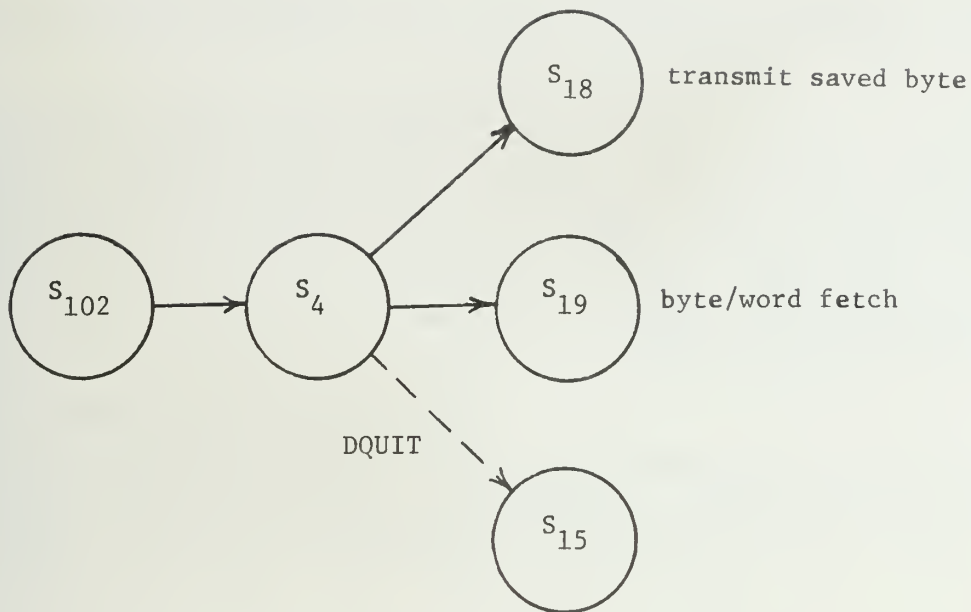
4.13 Wait for device to reset, S5A

Without this state the DSETA signal from S5 would be followed immediately by the DSETREQ in S0A, and some devices might not have reset their active bits in time to cease making requests. State S5A merely provides a delay.



4.14 Write state, S4

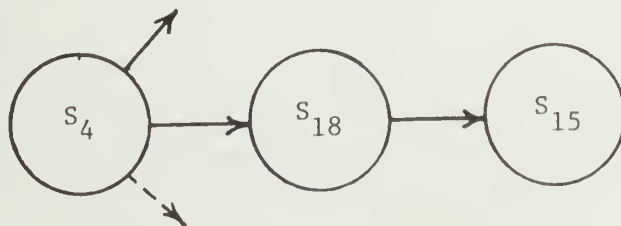
S4 performs the same function for writes as does S3 for reads. Two following states are selected, state S19 for byte or word fetch from memory, or state S18 for transmitting a previously saved byte to the device. If DQUIT is present, execution proceeds immediately to S15, bypassing data transfer. In addition, a flipflop S4HOLD is set (not reset until state S5) which causes the bidirectional character data bus to the devices to be outputting a character rather than receiving one. This prepares some data routing even though the actual data may not be available yet.



4.15 Transmit saved byte, S18

State S18 is entered if the buffered byte saved in the device registers is to be output. The main function is to set another flipflop (switch) called MUX9CTRL (reset by state S5) that prepares data routing between the saved byte and the character output transmitters on the device bus. State S4 has already caused the transmitters themselves to be energized. State S18 also causes the transmit register to be loaded. See figure 2.

The save bit is, of course, also reset by state S18 so that the saved byte is not resent over and over. The address and length are also suitably updated as in, and with the same restrictions listed for, state S24.

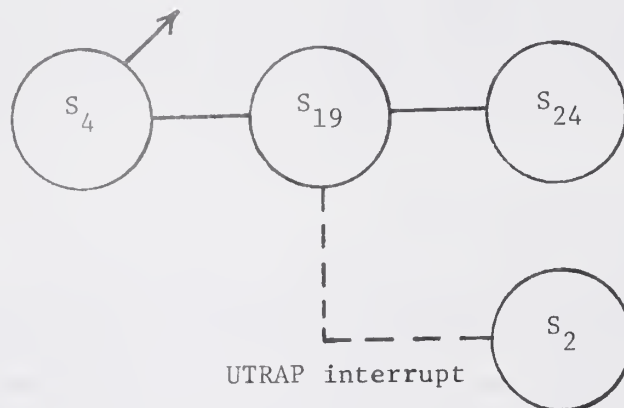


4.16 Fetch from memory, S19

When no saved character is available, a word is fetched from memory. If only a single byte is required from this word (as when, for example, the transfer is begun on the second byte in a word), the appropriate byte is extracted and loaded into the transmitter register.

If both are eventually to be transmitted, the low order byte is loaded into the transmitter register, and the high order byte into the save byte in the device registers. See figure 2.

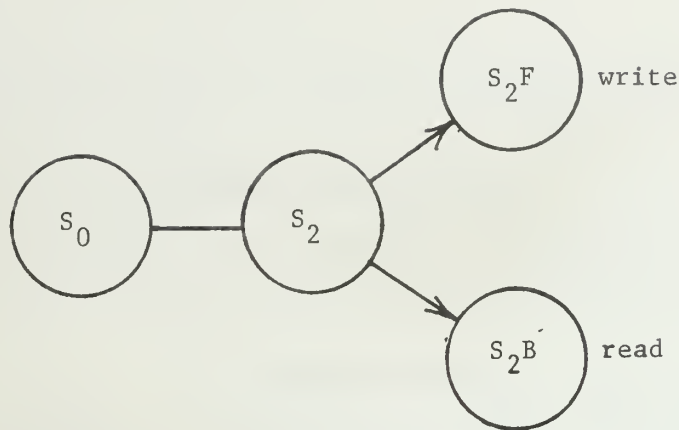
State S19 also sets MUX3ACTRL and MUX3CTRL which control loading of the save byte from the computer bus rather than from its other source, the device bus (used in read operations). The control flipflops are not reset until state S5 to avoid a race condition as the transmitter register and saved character register are loaded exactly at the end of state 19. The saved character register is part of the per device registers; MUX3 controls loading from either the solid state memory or the output of multiplexor MUX3A.



4.17 Register operation, S2

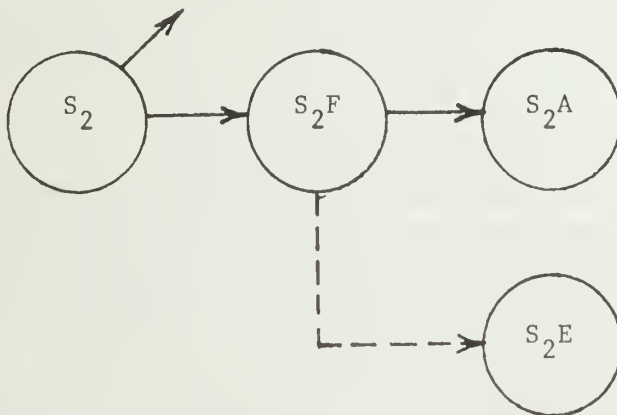
When an access by the computer to the device registers is detected in state S0 or via a UTRAP from states S11, S13, or S19, control proceeds directly to S2. State S2 is largely similar to state S1, loading the device registers with the information on the device selected by the computer. The same signal loading the device registers is also sent to the priority select matrix to load the device number register contained therein. Loading of this register from the computer bus address lines rather than from the output of the priority select matrix is controlled by a signal called DOUTREQ. This signal is generated by a flipflop set by S2 and reset by SOA. DOUTREQ also enables the selection matrix to select the device whose number is in the device number register.

Control proceeds after S2 to either state S2B for a computer read of the device registers, or S2F for a computer write into the device registers.



4.18 Load register from CPU, S2F

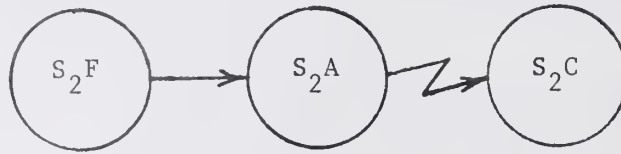
The state signal S2F causes either the first or the second word of the device registers to be loaded from the computer bus data lines. Which word is loaded is controlled by one of the address bits (bit 1) of the computer bus.



4.19 Register write, wait for completion, S2A

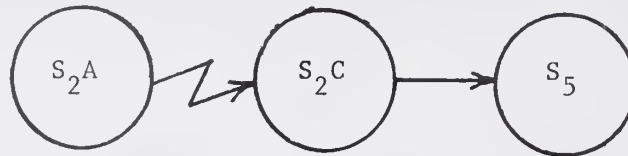
The register write is completed at the end of state S2F. Immediately, state S2A sends a slave synch (SSYN) signal back to the computer indicating completion. Slave synch is held until master synch, which originated the transaction, disappears. Master synch disappearing causes an abrupt transition from state S2A to state S2C, rather than by a normal clocked synchronous state transition. The synchronous clock is also disabled during the wait.

State S2A also resets the DATIP flipflop set by state S2B, thus signifying the end of a read-modify-write cycle.



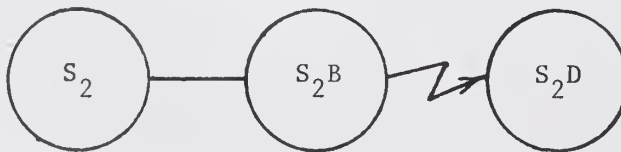
4.20 Register write, completion detected, S₂C

This state serves little purpose except to participate in the abnormal state timing mentioned for state S₂A. Execution proceeds normally from this state to S₅ to store the modified device registers.



4.21 Register read and wait for completion, S₂B

The action of this state is nearly the same as that for state S₂A. For a read, however, the computer interprets slave synch (supplied by this state) as also meaning the data to be read is available on the data lines of the computer bus. State S₂B is also included in the control for multiplexor MUX7A which controls which data is routed to the UNIBUS data lines. The control is further modified by the signal UFIFO which allows reading the FIFO in state S₂B if the appropriate address is referenced by a byte instruction (see table 12). Execution proceeds from S₂B to S₂D analogously as it does from S₂A to S₂C.

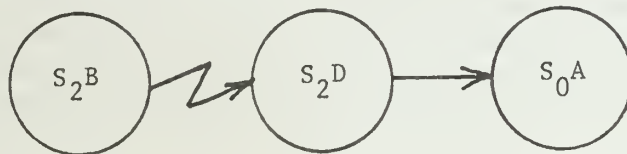


A flipflop called DATIP is set if the computer read is the start of a read-modify-write cycle wherein the computer expects no modifications to be made to the device register just read. Consequently, DATIP prevents the multiplexor from honoring any device requests, achieving the desired effect. This means a transition from state S₀ to S₁ is

temporarily impossible. DATIP is reset by S2A. The delay between the read and write cycle is so short that it is unlikely that any device request could be honored anyway.

4.22 Register read, completion detected, S2D

State S2D performs the analogous service for state S2B that S2C does for S2A. Since during a read operation the device registers are not modified, execution bypasses state S5 and proceeds directly to S0A.



4.23 Transfer leftover byte, S2E

Writing into the second device register can cause state S2E to start up independently and simultaneously with state S2A if a read transfer is being forced to terminate by rewriting the active bit with a zero, and if a buffered save character were being held ready for transfer to memory. State S2E stores that character if possible immediately after the computer releases control for the present register write operation. As state S2E must be terminated by state S0A, storing of the save byte relies on the computer not immediately attempting to access the device registers again. Only a short delay of one or two instructions is necessary in the computer program in order to honor this stipulation. If the stipulation is not honored, the last byte is discarded.

While the access restriction imposed for state S2E may seem to reveal poor planning, it is actually quite reasonable. State S2E is invoked only when the multiplexor is being used with an elaborate software system, and then typically only in a few well controlled situations, where the timing consideration may be easily observed. To avoid the restriction would require a very elaborate queue to retain data, which would then be stored when it became possible. When programming in a high-level language (such as in our UNIX system), it is almost impossible to violate this timing restriction anyway.



It is interesting to note one of the effects of continuously monitoring the device registers during a data read transfer. The length register is correctly decremented at the receipt of each character from a device, but the data does not appear in the memory buffer until the receipt of each second byte or until the transfer is stopped.

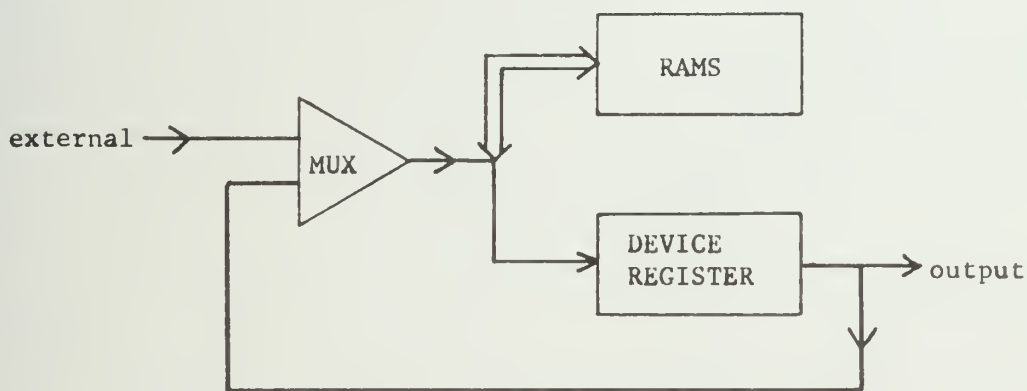
5 Data Registers

5.1 Data paths, RAMS

One of the basic concepts used in this multiplexor is that of storing the device registers for each device in solid state memories called RAMS. In order to perform any service for a device, whether initiated by the computer or the device itself, the appropriate forty-one bit word in the RAMS is loaded into a set of (device) registers.

Different fields within the device registers have different connections according to their purposes, although there are great similarities. See, for example, the save byte used for character buffering, Figure 2. Figure 3 indicates many of the data paths corresponding to earlier descriptions under "Functions and Capabilities." MUX1 allows loading word one of the device registers from solid state memory or the computer bus; MUX2 likewise for word two.

The RAMS used here employ a tristate bus for packaging reasons with input and output on the same wire. Forty one tristate multiplexors are used, one with each device register bit, as follows:



Note that this composite system always allows two-way communication between the device register and RAM, and allows the device register to be loaded from one external source. The only path to the RAM is through the device register.

5.2 Slow access to by computer

The program-like nature of the states controlling the multiplexor results in a relatively slow response to outside requests. This means the access of a device register by the computer takes 2 or 3 microseconds rather than a more conventional one-half microsecond found in simpler devices. In an interrupt driven DMA environment, however, such accesses are relatively infrequent, so this is not a significant disadvantage. Care was taken that DMA cycles are as fast as computer bus protocol permits.

Slow execution of states also has the interesting effect of lowering the maximum repetitive transfer rate of all devices combined. While still quite high (two or three microseconds), the transfer rate is low enough to avoid causing disk or drum overruns, an attractive feature.

5.3 UTRAP, shared resources concept

For simplicity the computer is not permitted to do a device register access at the same time that the multiplexor is honoring a device request. To allow the two to occur at the same time would require separate device registers for the two functions. This in turn would create the possibility of two different copies of the registers for one device, only one copy being retained (the one stored last in the RAMS). The use of the device registers as a modifiable resource must, therefore, be serialized.

Serialization creates a new difficulty since the computer bus cannot be shared. If the multiplexor decides to perform a memory transfer (state S11, S13, or S19) just as the computer attempts to access a device register, the multiplexor will be deadlocked attempting to acquire control of the computer bus. The computer, which we are assuming has control, will be deadlocked also waiting for a response. In a busy timesharing system, this situation while relatively unlikely can still occur every few minutes. In special purpose stand-alone programs (test programs, bootstrap programs, etc.), a frequent technique is to examine a device register in a loop rather than rely on interrupts. In such a program, the deadlock is wholly unavoidable.

Avoiding this problem was the most difficult of the early design considerations. It was resolved by allowing a device register select to cause execution of states S11, S13, or S19 to be abandoned and execution to begin immediately at state S2. This constitutes a form of interrupt. Because execution is abandoned (and DRESET not issued), the device will still be making its request which will, therefore, be honored normally at a later time as though it were a new one.

This strategy implies that no action be taken until after states S11, S13, or S19 unless the action may be repeated. Echoing is therefore decided upon, and a flipflop set to remember the decision, in state S3. The echoing is not performed until the address is incremented in states S24, S12, or S18. Modifications to the device registers prior to an interrupt are unimportant as the registers are never stored back into the solid state memory (state S5 is not executed).

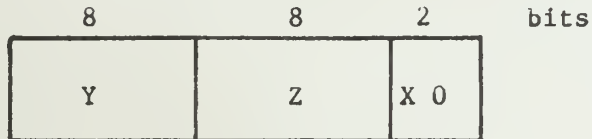
While the UTRAP concept was quite troublesome initially, it did not create much additional circuitry or make the multiplexor design excessively complex.

The basic functions performed by UTRAP are restarting the synchronous state clock, resetting states S11, S13, and S19, and causing

execution to begin at state S2. The UTRAP flipflop is reset by state S2 having started successfully, and also by state S2E. State S2E performs memory transfers, but only if convenient. Of necessity, state S2E cannot delay, and UTRAP is therefore held reset, not permitted to occur.

5.4 Addressing

The current scheme is to give each device register for each device its own address. The address structure is as follows:



As each device has two words for its registers, the single bit X is used to determine which is being accessed in a given reference. Z is eight bits, allowing 256 different devices to be separately addressed. An option allows this to be changed to six bits for 64 devices in order to use less address space for device registers, although there seems typically to be little need to do so. If Z is odd the device is an output device. Y may be established by some changeable wires in the multiplexor, allowing the 1024 bytes of device registers to be relocated in the computer's address space. If 64 devices are in use, the top two bits of Z may be set as well, and the address space required is 256 bytes. By convention the device registers of all devices are usually placed somewhere in the highest 8192 bytes (4096 words) of memory, but this still leaves some choice in the value of Y.

An alternative addressing scheme is discussed in the chapter on new proposals.

A special piece of hardware detects byte-only read references to location:



In response to the read, the byte with the address of an interrupting device is supplied from the hardware queue of such device addresses. The readout from this queue is destructive. Such reads from this location should, therefore, be in one-to-one correspondence with interrupts. If the queue is empty, the address of the most recently interrupting device is supplied.

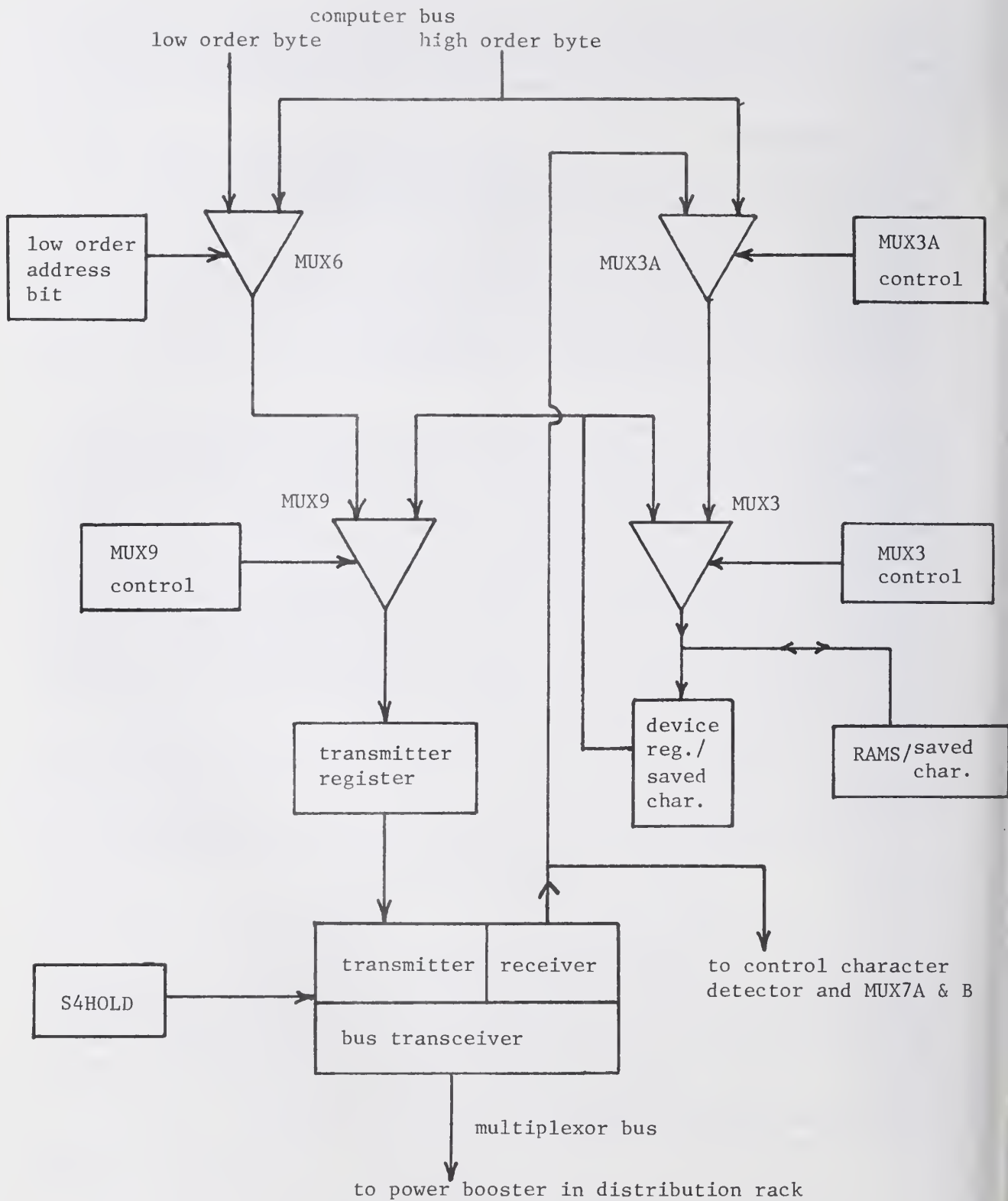


Figure 2. Data flow of transmitted character

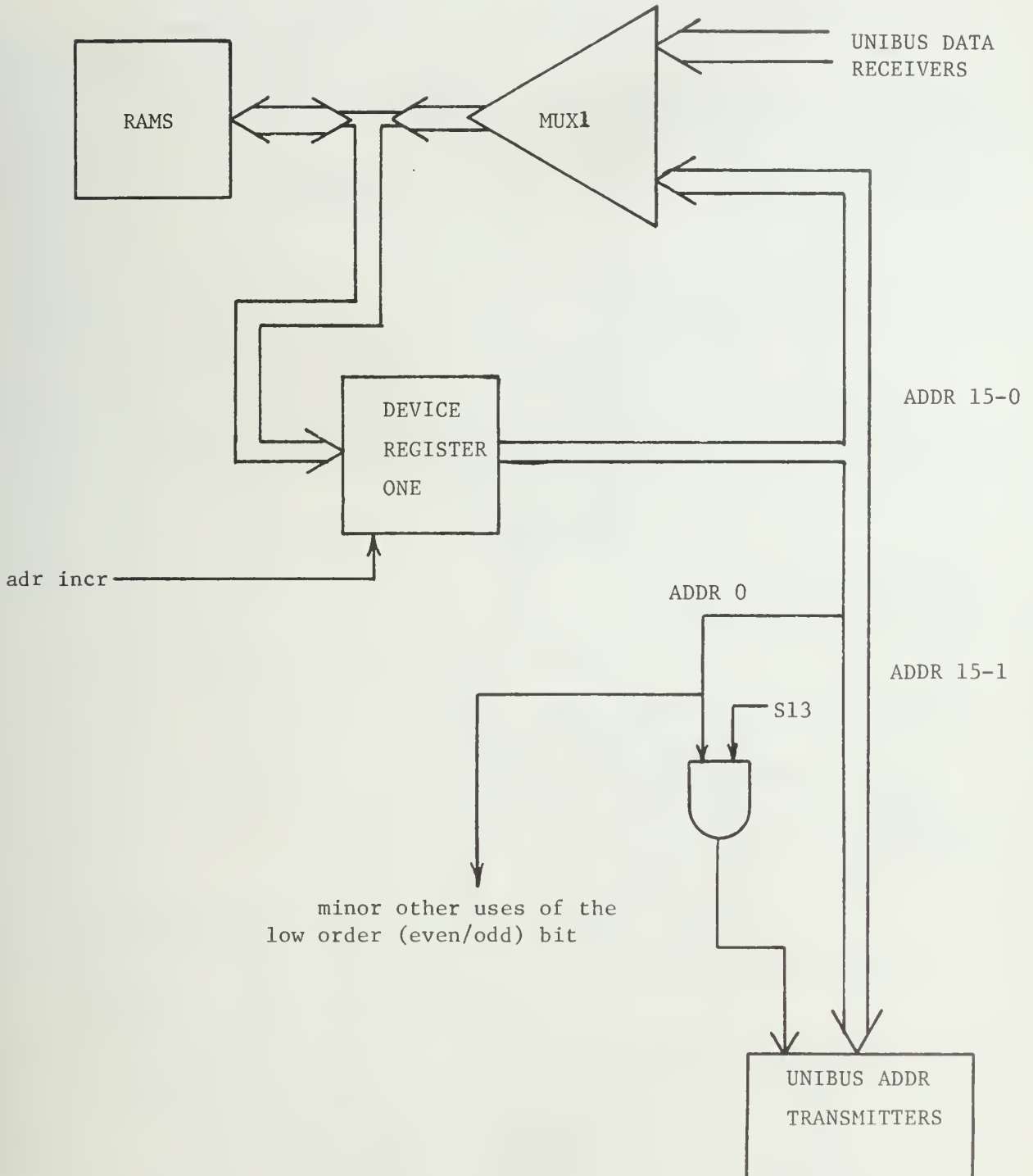


Figure 3. Data flow of first device register

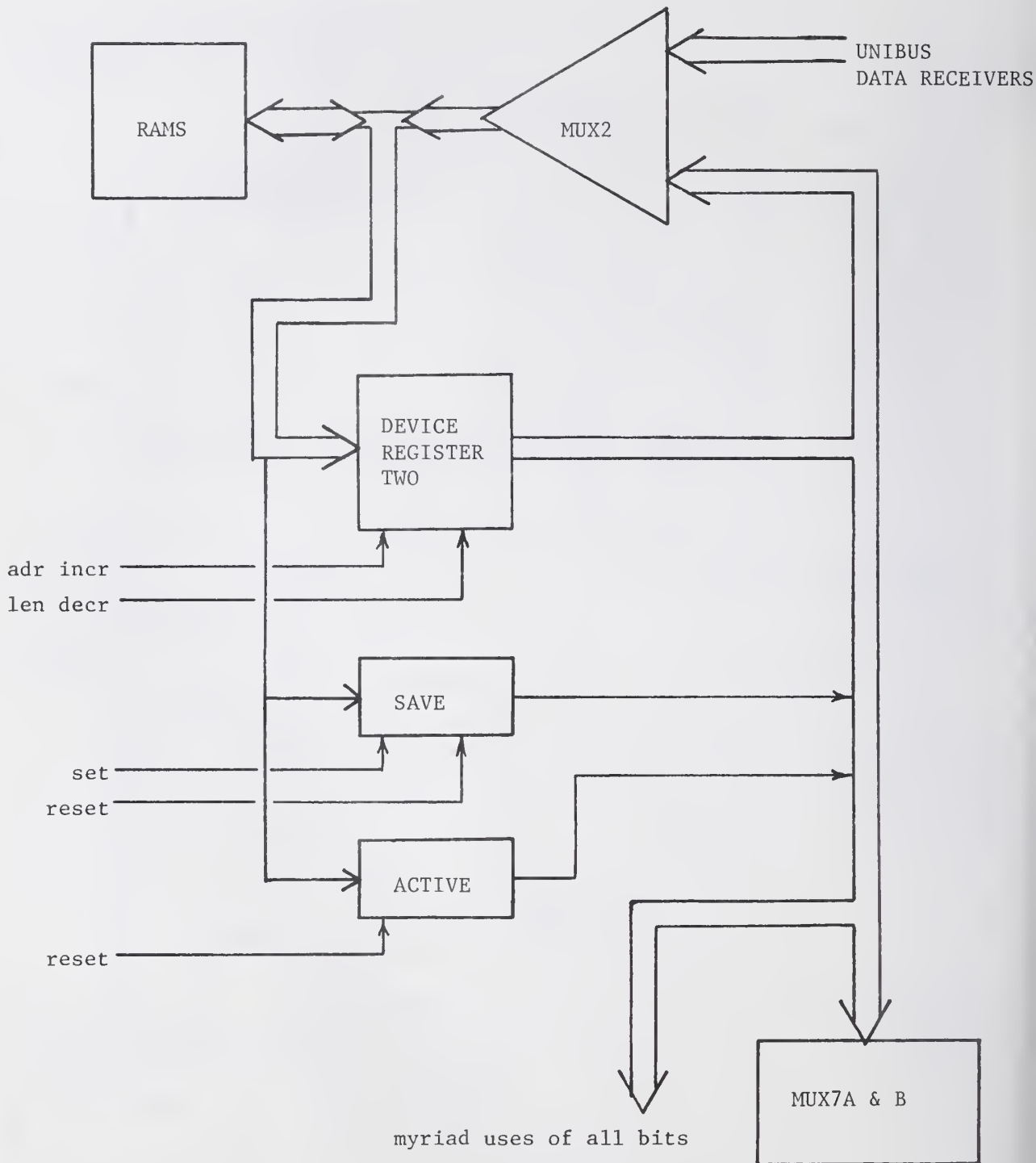


Figure 4. Data flow of second device register

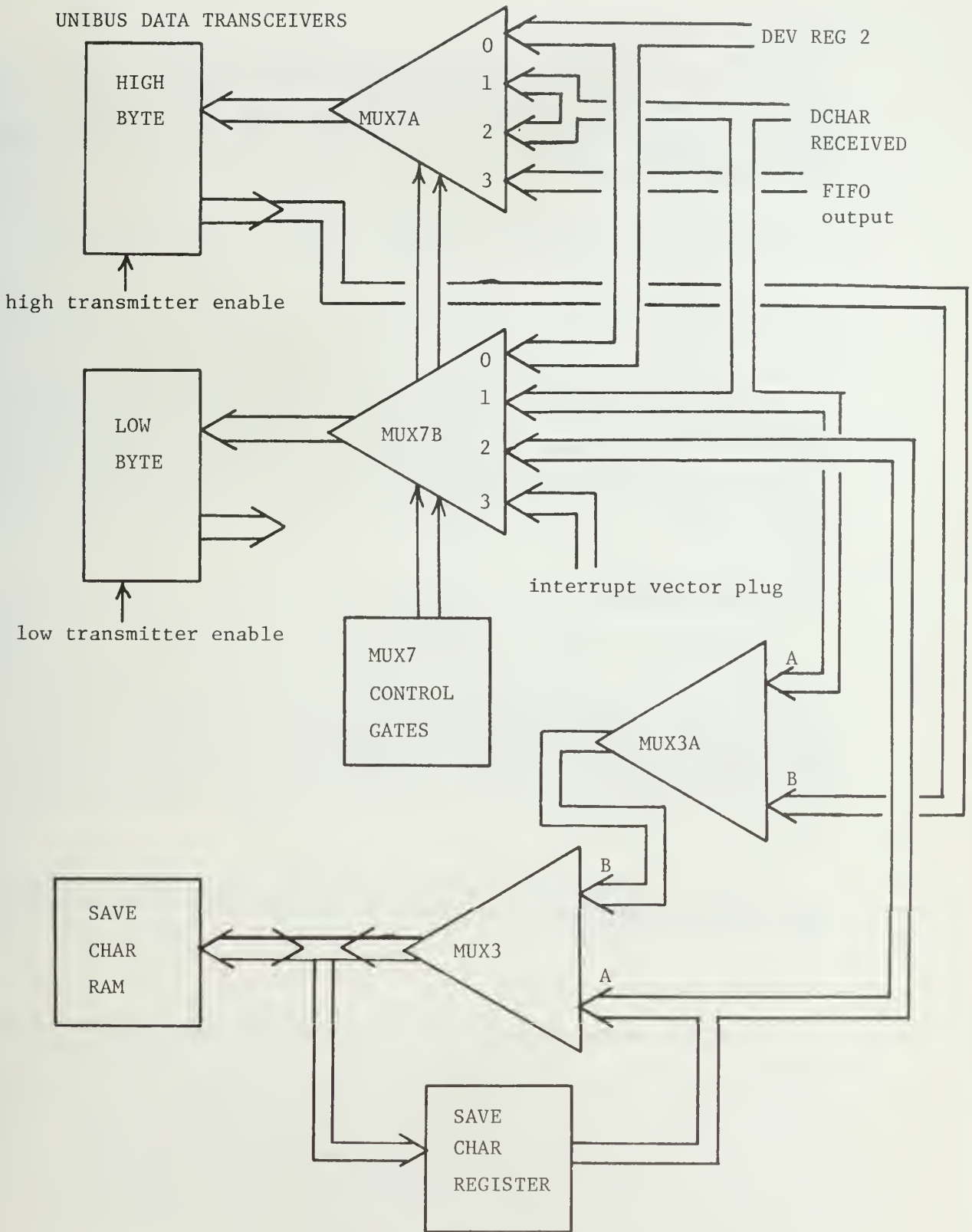


Figure 5. Data flow of received character

6 Device Control Signals

All outgoing control signals are always sent to all devices. Each device is supposed to gate these signals with its select signals (both vertical and horizontal) so that only the selected device responds. The number of control signals is kept small. They are DSETREQ, DECHO, DSETA, DRESET, DZAP.

Incoming control signals are all similarly bussed, and must be given only when the device is selected. These are DQUIT and DFINAL.

Other control signals for row and column request and select are handled differently and are discussed in the section on the priority matrix.

Timing on all control signals is very generous in order to eliminate all possible race conditions. Signals are typically not adjacent, being generated by nonadjacent multiplexor states, or are adjacent in a noncritical situation. The CTRAP function is implemented in the main multiplexor unit and hence there is no associated device control signal. Table 2 lists all the control and data signals on the device bus.

6.1 Set request

The DSETREQ signal is unusual in that it is to be honored immediately by all devices rather than only when the device is selected as for the other outgoing control signals. This signal does not take part in the data transfer cycle; it is used to tell devices that they may commence requesting or cease requesting data transfer. It is the state signal SOA, amplified. In each device a flipflop or buffer is loaded with the binary value of the devices' desire to communicate or not. Transitions of intent are therefore visible to the priority matrix only when this signal is issued.

Obviously the multiplexor does not use polling. Such a technique is an extreme bottleneck, and would slow down the operation by more than two orders of magnitude.

6.2 Echo

The DECHO signal is used only by interfaces which employ hardware automatic echo. The signal is produced during a read operation (as requested by the device) just before the reset signal. It is not issued if the received character is a control character. Typically, a terminal interface will use this signal to activate its transmitter device which will then transmit the character on the device bus data lines supplied by its own receiver device. Obviously, if an output transfer is active on this transmitter, some data may be lost. One should be able to see that the implied restriction is reasonable.

6.3 Set active

The DSETA signal causes the selected device to load a new active status from the DACT line and new option bits from the DOP1 and DOP2 lines. The latter two may, of course, be ignored if the device has no options. The data will usually represent no new information as the DSETA signal is generated at the drop of a hat. Devices should, therefore, be sensitive to the value of these three items rather than to the fact of their being recently loaded.

Devices should make no requests when their active bits are off.

6.4 Reset

The reset signal DRESET denotes the successful completion of the device's request. For a read operation it means the character has been removed from the device bus data lines and has been or will be transmitted to main memory in the computer. For transmit operations, DRESET indicates the presence of the desired character on the device bus data lines ready for the transmitting device to remove and store it.

6.5 Zap

This signal (DZAP) is generated whenever the computer is being initialized (INIT is present on the computer bus). Devices may use this signal to reset themselves if they so desire.

6.6 Quit

A special feature of the multiplexor uses the DQUIT signal. When a device makes a request, if this signal is present, no data request takes place, and the transfer is completed, the computer being appropriately notified.

6.7 Final

Similar to DQUIT is DFINAL. This signal also indicates completion of the data transfer, but a byte of data is transferred during the request cycle in which the DFINAL signal is seen.

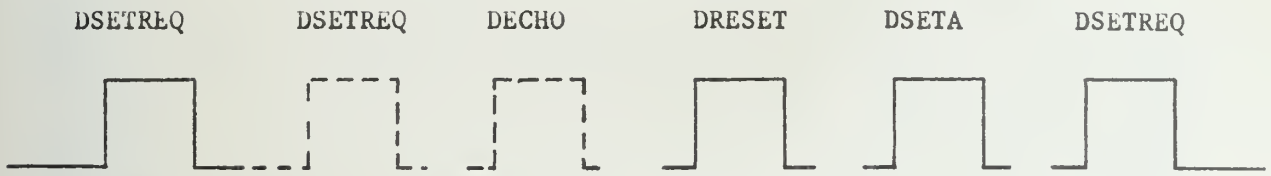
7 Protocol at Devices

Priority arbitration occurs in state SOA. The state signal SOA is equivalent to DSETREQ. On the leading edge of this signal, each device is supposed to set its request flipflop if it chooses to perform a cycle. A device formerly doing a data cycle which is now finished should clear its flipflop. The request flipflops are used by the device to drive row and column request lines to ground (see the chapter "Priority Matrix"). These lines are translated into an eight bit device number of the highest priority request in the distribution rack by a set of priority encoders. The device number register is being continuously loaded from the priority encoders. A settling effect takes place, due to cable delays, transition times of request flipflops, delays in the priority selection matrix, etc. This activity limits the shortening of the duration of the multiplexor states, although it is not the only limit. State SO merely waits for the now stored device number register output to be carried through the multiplexor bus to the main multiplexor unit.

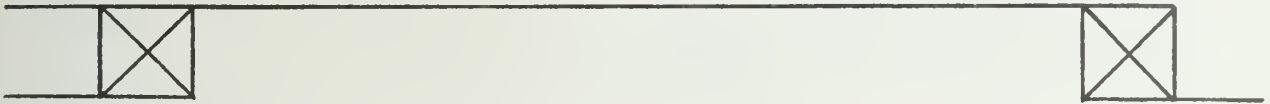
Figure 6 shows the typical timing diagram for device control signals. For the selected device, at the end of DSETREQ, the row and column select signals will be asserted (zero volts on the device bus). DSETREQ may be repeated if the multiplexor is interrupted during its data transfer (see UTRAP). Sometime later the DECHO signal will be issued, if appropriate, then DRESET to note the completion of service. Options and active bits are reissued then (DSETA). Only the active bit could have changed, going off if the transfer is ending.

The effect on request flipflops contained within each device may be noted as well. Even though DRESET clears the request internally (such as resetting a UART receive flag), the condition is reflected in the request flipflop only at the next DSETREQ.

Signals on the four main device bus control lines



Column select to selected device (not inverted)



Row select to selected device (not inverted)



Request flipflop within non-selected device wishing service



Request flipflop within selected device



Figure 6. Timing of main control signals to device.

8 Priority Select Matrix

8.1 Design choices

The priority select matrix is one of two ways to solve the select/request problem. In some ways the other (linear) method seems preferable. With the matrix each device has to worry about both a row and a column select, and both a row and a column request, rather than a single wire for each. The alternate (linear) method would require a special circuit card in each device rack to handle linear priority selection for that rack. This has the advantage that bus receivers could be located in this special circuit card for device bus signals supplying that rack, rather than being duplicated within each interface card. Proper termination resistors could also be added to this circuit card, probably allowing the device to operate faster. Speed of the device bus and speed of the RAMS are the two factors influencing the speed limitation of the multiplexor.

Counter considerations are that the multiplexor was faster than needed by far, an extra circuit card per rack seemed to me unaesthetic (a purely personal reaction), and the circuit card would force the number of interfaces per rack down from 15 to 14. Having the device bus cable enter the rear of this special circuit card would have prevented losing an extra interface slot, but this would not have been feasible with components available.

As is typical for many multiplexors, most of the cost of this multiplexor when many devices are being interfaced is in the per-port add-on cost, specifically in the interface cards. While the cost of this unit is less than that of commercial units of vastly less capability, the cost probably could be lowered still further by simplifying the devices through the linear priority selection scheme.

Nevertheless, I chose the matrix scheme, and it worked out well. Indeed, it has a desirable simplicity and lack of complications that was preferable in a prototype construction.

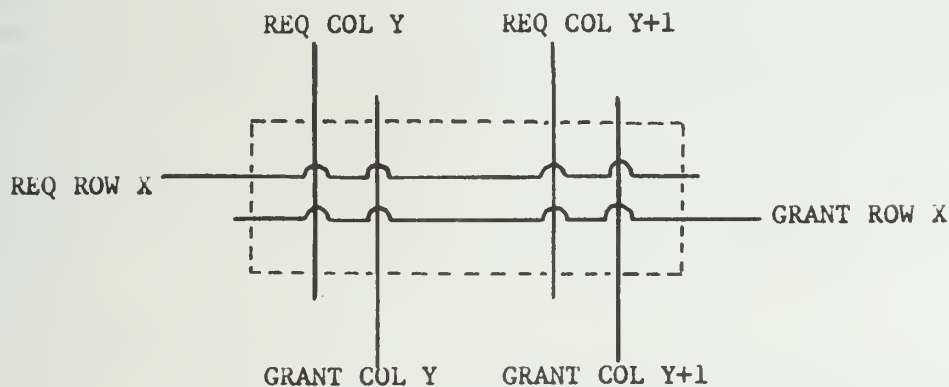
8.2 Matrix structure

An interface may perform two basic functions on its own initiative: read and write. A typical interface contains two logical devices. The logical port assigned to an interface is comprised of two device numbers. A read is a data cycle by the even device and a write is a nearly identical data cycle by the odd device.



Logical composition of port 20

An interface appears in the priority matrix as follows:



Both the read and the write device are in the same row. In this drawing, Y must be an even number. The port number would be $X * 8 + Y/2$ (see Table 5). The two device numbers are $X * 16 + Y$ and $X * 16 + Y + 1$ (see Table 3). As may be seen from the tables, there are 256 devices arranged in a 16×16 matrix (table 3) and therefore 128 interfaces may be accommodated.

To perform a read, the REQ COL Y wire is driven to ground (a zero voltage applied). A resistor usually keeps the voltage, on this wire, at five volts. During the appearance of the GRANT COL Y signal, the device may drive REQ ROW X to ground. The GRANT COL Y signal may be withdrawn and then reissued if a higher priority device appears. The device must issue the row request only while column grant is present. When both GRANT COL Y and GRANT ROW X are present, the data cycle begins in the multiplexor. At this point the device gates its eight bit byte of data onto the bidirectional data lines. The signal DRESET from the multiplexor indicates the cycle is over, and the request may be abandoned.

A write cycle is identical except that data is taken from the bidirectional data lines rather than being put on them, and except that it is performed by an odd numbered device. In the above figure, column $Y + 1$ and row X are used for the write request cycle.

The request is handled in the priority matrix by priority encoder

integrated circuits. The lowest numbered column is selected first, then of the row request signals thus activated by the column grant, the lowest row number is selected. The output of the priority encoders is an eight bit device number. This number, or one from the main multiplexor unit may be loaded into a register controlling the grant circuitry. A device may therefore be selected directly, ignoring outstanding requests. This feature allows the computer to communicate option bits and the active bit directly with the devices (state S2 and subsequent states).

The priority of devices can be used to advantage. If one wished to build a byte parallel device for communicating with another computer, one could construct it to operate asynchronously as rapidly as possible, and plug it into a low priority slot to avoid blocking other devices. Despite the matrix design, a strict linear ordering of priority exists.

9 Software Compromise

In this section, some decisions involving both hardware and software design are noted briefly. In some cases, the hardware design was compromised for the software. In others, the reverse is true.

9.1 Definition of ASCII Control Characters

The most significant compromise is the definition of ASCII control characters. The CTRAP option uses the octal codes for ASCII control characters to stop an input data transfer, and the same control code definition is used to decide whether or not to echo. More flexibility could be obtained by using a microcomputer inside the multiplexor at the expense of speed, and with a probable increase in cost. The CTRAP feature makes a very convenient way to send variable length blocks of data back and forth with a foreign computer which possesses only a serial interface. This requires that the data be encoded to avoid control characters before the end of the block. This exact scheme is now in use. Another implication of this compromise is that programs desiring to perform some action on receipt of a printing ASCII character must cause reads of one character at a time to be performed.

9.2 Delayed data store

As noted elsewhere, due to the buffering of characters into words, the length register may be examined and the deduction made that a character has been received, although no character has yet been placed in memory. If, however, the computer forces the transfer to stop at this point, the saved character will be stored into memory.

9.3 Delay before echo

A peculiarity arises from the type of UART used on our device cards, and the way they are used. It has nothing to do with the multiplexor design. A UART (Universal Asynchronous Receiver Transmitter) is a single integrated circuit that performs serial/parallel conversion in both directions. Because on output the UART buffers up to two characters the transfer appears to be done too early from the computer's viewpoint. If automatic echoing is then enabled and a user types a character, the device will fail to output through its busy UART the character typed, and it will not be echoed. Although this is a low probability event, it will occur occasionally.

My solution is to delay activation of hardware echo for one-fifth second (two character times of the slowest device, a teletype) after any data is output. Since echoing must be done by software anyway while output is sent to the terminal, this merely extends the period by one-fifth second. Because of some laziness in my implementation, if one types fast enough, an interesting (though not offensive) jerkiness occurs in the echoing. A human cannot type fast enough to sustain this effect for more than a few characters.

The UART provides a completion signal independent of this buffering, and my interface could have used this signal instead (in fact, I believe I have some of each type). This is not as desirable, however, since the early completion signal gives the UNIX timesharing system some time to get another buffer of data ready to send, thus smoothing out the apparent flow of data. At 9600 baud this assistance is quite useful since the UNIX system does not respond rapidly to interrupts.

10 Data Transfer on UNIBUS

Bus control on DEC devices is typically handled by a DEC M7821 bus control card which manages both data transfers as bus master and interrupts. Willing to take on any challenge, I have designed my own circuits for this purpose. Mine are perhaps marginally faster. The M7821 is not as fast as bus protocol allows; however DEC processors are not as optimal and asynchronous as they might be either.

The exercise of building my own bus control was nevertheless necessary as I needed one special ingredient: the ability to cleanly have a change of heart and cancel an attempt to do a data transfer cycle. This need comes from the special nature of the device registers in the multiplexor. Since they are located in RAMS (solid state memory) and a single holding register exists to access the registers for one device at a time, a scarce resource is created with two competing factors. The multiplexor may decide to transfer data on the UNIBUS and be locked up (e.g., in state S13) waiting for access to the UNIBUS. At the same time, the multiplexor may never be able to gain control because the processor has bus control and is attempting access to a device register. The multiplexor can not handle a device register access in state S13. This problem is discussed elsewhere in this report. A simpler device has no trouble since device register accesses are handled by a simple set of independent circuitry. The device registers in a simple device are always accessible in some TTL register.

The bus control circuitry, despite the difficulty of design, is straightforward and not complex. For the most part it implements meticulously the bus protocol described in the DEC peripherals handbook [1].

One fine point of protocol is not forcefully presented in the DEC document and caused some grief. The solution of the problem is due (alas) not to myself but C. W. Gear. The difficulty arises from a passage with more than one interpretation. At first glance, bus control appears to be obtained by the bus having been successfully arbitrated for (via NPR and NPG) and bus busy (BBSY) being off. The processor may however decide to do another data cycle. The gimmick used to indicate this decision is to require waiting after the arbitration grant (NPG) until NPG is removed. If the processor decides to swipe an extra bus cycle, it continues to assert its arbitration grant (NPG) until some time during the last BBSY cycle. A peculiar but working protocol. This makes the operation of the processor as bus master different from all other devices, which are supposed to hold BBSY continuously to perform multiple data cycles.

The two main input control signals for bus control are REQ (an OR of S2E, S11, S13 and S19) and UTRAP. REQ is asserted by any state wishing to do a data cycle. The cycle then takes place automatically. UTRAP is very simply the address decoding of an access to the device registers; it resets everything in sight in the data transfer logic.

UTRAP can occur during some portion of arbitration but never in the middle of an actual data transfer, as the processor (or some other bus master) is clearly itself in a data transfer, trying to access the device registers. Hopefully no one would ever initiate a multiplexor data transfer into its own device registers.

The bus control circuitry emits a signal called TMASTER which controls gating of information onto the address and data lines of the UNIBUS.

The remainder of the data transfer logic is understandable from the protocol and needs no further illumination.

11 Interrupts

The interrupt bus control logic is a much simpler version of the data transfer bus control logic. Incoming request signal IREQ is driven by the FIFO "data available" signal. There is no other connection to the FIFO. IMASTER controls gating the interrupt location onto the data portion of the UNIBUS. This location is a full sixteen bit address. Interrupts are completely isolated from any connection to other multiplexor circuitry except for the IREQ signal. There is no interaction. Given the bus protocol, the bus control circuitry is completely transparent.

Interrupts may be disabled by setting the processor priority too high (such as priority seven). In addition, a RESET instruction or the depression of the front panel START key will reset the multiplexor and clear all data transfer active bits. Programs which do not use the multiplexor are therefore indirectly guaranteed freedom from interrupts. Interrupts may not be disabled during operation (there is no device register for the multiplexor as a whole) except by processor priority manipulation. This scheme has not been particularly troublesome. Standalone test programs may simply set priority seven and wait for transfers to finish by looping on the active bit in an individual device register.

12 Error Lights

Two error lights are mounted on the printed circuit cards of the main unit. One, on the control card is for indicating that state S101 has been executed since the computer was last restarted or powered up. This typically means that a device does not correctly realize that its active bit should be off and is making requests. The other light on the data board indicates that the FIFO has overflowed at least once since the computer was restarted or powered up. Neither condition has ever been observed to occur except when testing the correct operation of the error detecting circuitry.

13 Automatic Echo

As previously mentioned, data transfer is direct to memory. The advantages of DMA (direct memory access) are well known and are of great benefit for any kind of output and for input from other than terminals. Use of DMA input from a computer terminal, however is of more questionable value. The difficulty is that the multiplexor must echo the typed characters (all civilized systems being full duplex). There are reasons why this added complexity may not be worthwhile. Although the overhead for interrupting on every character, especially in a large installation, is certainly noticeable it is not overwhelming. An echo control signal has been made available to each device pair by the multiplexor for hardware echo. A very slight additional amount of hardware is required in each interface using this option to utilize the signal. The extra hardware, while not extensive, is still a negative factor associated with automatic echo. This feature could be moved into the multiplexor proper (see Proposals).

The worst complications arise in software control of the hardware echo feature. It is generally not appropriate to automatically echo control characters, and the echo control signal is therefore not generated for these characters. Fortunately, control signals are already being tested for in the multiplexor, and not echoing them is not difficult. The software, however, must be aware of this further complication. It is not useful to employ automatic echo without the control trap feature to allow the computer to gain control and decide whether software echo or some other action is appropriate. Other software difficulties include some strange timing considerations discussed in the chapter about software peculiarities, and the fact that echo must be disabled anyway and done later by software while normal computer output is being sent to the terminal.

To put forth the effort in hardware and software that hardware echoing required, for a single not very large installation, was probably worthwhile only in the name of research. Without this excuse, and foreseeing the complexity, it would not be worthwhile. Having already gone to the trouble of solving the difficulties and developing the techniques required, automatic echo is desirable marginally. It is more efficient, but the code is more complex, and therefore larger. This type of code is usually required to be resident in the computer, making its size an important issue.

14 Proposals

Inevitably, after construction, there are second thoughts and wishful thinking. All improvements that were thought of were added (and are described in this report) except in the area of device register changes.

The current arrangement gives two words of address space to each device, thus taking up a large amount of space for 256 devices: 1024 bytes. This is only 12.5% of the available device register area, which is good considering that 128 interfaces can be handled with it. Still, it would not be practical to enlarge the per device register area beyond two words, and this limits expansion.

As it happens, however, an expansion could be desirable, and would not be particularly costly in multiplexor design or parts. A better arrangement might be:

174000	IDN	EMPTY
174002	ADR	
174004	LENGTH	
174006	CURRENT DEV	ERRORS
174010	SENSE	OPTIONS

In the new scheme, ADR and LENGTH correspond to the two words currently used for each device, except that the option bits could be eliminated from the LENGTH register (or retained if desired) because of the new separate field at 174010 (these addresses are for illustration purposes only). The registers being examined by an access to ADR (174002) or LENGTH (174004) are those for the device whose number is in CURRENT DEV. CURRENT DEV may be changed to any device number. IDN is the interrupting device number similar to the previous facility, popped from a FIFO each time it is read. An EMPTY bit could be added to indicate that the FIFO is empty, a convenience for test programs or special purpose programs, though not strictly necessary.

The ERRORS field would allow examination of the error conditions

now indicated only with indicator lights (FIFO overrun, S101 entered). In addition, this field could contain interrupt enable bits for the whole multiplexor, a feature not currently available, though not particularly important.

The SENSE field would allow reading of a full eight sense bits (through the DCHAR lines on even numbered devices only is one possibility). The OPTIONS field would allow a full eight options to be set (through the DCHAR lines on odd numbered devices only for the analogous possibility).

Clearly an extra word can now be added per device (like ADR and LENGTH and SENSE/OPTIONS) if more features are to be added to the multiplexor. Even without adding another word, the option field of LENGTH could be freed if the new SENSE/OPTIONS feature was added and the old scheme with two option bits and two sense bits was abandoned. This would allow addition of multiplexor implemented functions.

Possibilities for new functions are

1. have the ECHO function be controlled completely in the main multiplexor.
2. selectable alternate definition of control character codes.
3. disable interrupts on individual devices (probably not too useful).
4. more option bits of the conventional type rather than using eight bits on the DCHAR lines. (preferable)
5. indicator bits indicating whether a transfer was stopped by CTRAP, QUIT, or FINAL.

The only feature I have ever really felt the need for was extended option bits, although this may be gotten around. One extra conventional option bit per device would be quite sufficient.

15 Kinds of Interfaces

In order to illustrate the kinds of flexibility possible within an interface, a brief description follows of some of the interfaces that have been thought of. Appendix D contains the complete circuit diagram of the first interface described (EIA, speed selectable), so that anyone trying to puzzle out the working of the device bus in detail will have a straightforward, specific example.

15.1 Speed Select EIA

Although at first we used only teletype interfaces, the most common interface on our system and the most commonly needed interface on most systems is a speed selectable serial interface with EIA level shifters. Three option bits are used to select speeds of 110, 300, 1200, 2400, 4800, 9600, 19200, and 38400 baud. The fourth option bit is used to control hardware echo. Serial parallel conversion is done by a UART. The data received (DR) and transmitter empty (THRE) signals are used for the internal read and write request signals, and are blocked respectively if their active bit is off.

15.2 Busy Interlock

The busy interlock interface resembles the previous interface except for having four EIA speeds selected by two option bits, with one option bit enabling the interlock. When enabled, the transmitter empty (THRE) signal may not be reissued after the first occasion unless the EIA signal BUSY goes on and then off again. This restriction applies equally between the last character of one output transfer and the first character of the next as it applies to adjacent characters within a single transfer. A timeout of a few seconds is provided to avoid hangups due to malfunctions in the equipment being served or programming errors.

This interface is useful for driving terminals with various kinds of delays such as for carriage return, form feed, storage tube screen erase, and other functions where a signal can be made available for use as BUSY. The Computek graphic display terminal which has extremely variable display times for different length vectors and different characters is ideally suited to this type of interface. The interface is also being used to drive several output only devices such as printers and selectric typewriters with somewhat erratic output needs due to their particular mechanical construction.

Commercial equipment vendors sometimes discuss a technique of raising and lowering request to send (RTS) in order to produce such an effect with their terminal equipment, which would be just what this interface is designed for.

15.3 Telephone Dialer

As it is quite convenient to dial the phone by entering a simple command at the terminal and have the computer look up the phone number and dial, this very simple interface has been built. Only one of the eight output bits is used, being loaded into a flipflop each time a one-shot expires, and makes a data request. The flipflop output drives a reed relay in the vicinity of the phone, whose contacts are in series with the dial contacts. This interface has almost no parts.

15.4 Clock

When a read transfer is initiated, the clock immediately provides the current value of an integer of four bytes, thirty-two bits. The integer is incremented once each second and may also be loaded by an output transfer of four bytes. The integer represents UNIX format time, the number of seconds since midnight January 1, 1970. A standby oscillator and rechargeable battery assist its accuracy.

15.5 Paper Tape Readers

A thirty-two by eight FIFO with a half-full flag happens to be a paper tape controller, at least given some thought. The photocells, via some bus receivers used as threshold detectors feed the FIFO inputs; the output feeds the device bus. The half-full flag operates the magnetic capstan drive, tending to keep sixteen bytes in the FIFO. The amount is perfect to handle uneven data requests by the computer and mechanical sluggishness in disengaging the capstan.

15.6 FIFO Card

This is a variation of the EIA interface in section 1 with a FIFO on the receiver side to avoid losing characters while a read transfer is not in progress. In addition, a timeout facility is activated when hardware echo is deactivated that uses the DQUIT signal to terminate the transfer when a break of more than 1.6 character times appears in the receipt of consecutive input characters.

This interface is intended for communication with another computer installation where the data is being received at high speed, but is completely unformatted on a character basis. One such interface for instance is connected to a regular terminal line on the CYBER 175 and permits any timesharing terminal on our UNIX installation to become a CYBER terminal by running a program that "connects" the two together. The timeout allows gaining the efficiency of block transfer DMA with the freedom of not having to know how much data is about to be received.

Naturally, hardware echo is always disabled, and is merely provided because it makes this interface interchangeable with the regular speed selectable version for use in terminal applications.

16 Conclusions

It is disappointing to note that computer manufacturers do not today provide a cheap vehicle for accommodating easily a large number of interfaces.

In most computer installations, the number of interfaces that should go on a simple device bus greatly exceeds the number that need to be attached to the main computer bus. When a large number can be connected simply, the improvement that occurs is not just one of quantity, but is a fundamental change of the quality of service that the computer can perform. Functions that were previously (and justifiably) considered frivolous, become inexpensive enough that there is almost no point in not doing them. This multiplexor appears to break a significant cost barrier that allows the minicomputer to become more directly involved with the total environment and the needs of those using it.

Another accomplishment of the multiplexor is to shift the bottleneck on the computer from terminal I/O to that of computing resources. Even on machines like the DEC 10 and the CYBER 175, used at this campus, system administrators severely restrict terminal I/O rates, usually to 30 characters per second, and always at least to 120 characters per second. By contrast, our PDP 11/40 can easily maintain 15 people in the editor or some other very low compute function at 960 characters per second without difficulty. Other of the multiplexor's ports may be simultaneously communicating with many other computers at speeds of 120, 960, and 3840 (approximately, this speed is not a constant) characters per second without impact. These capabilities are due to the obvious (but impressive) advantages of DMA I/O and low cost.

Nevertheless, the best value has been in flexibility. Any device requiring uneven data rates or even slightly unconventional control signals usually is a burden on the budget (to purchase a special, costly interface) or a burden on someone's time (to build same). The very simplified device bus of this multiplexor has made this kind of custom work the real strength of this unit.

Appendix A -- cost breakdown

What follows is a crude and somewhat out of date cost estimate for a multiplexor with 30 serial interface cards. It is not necessarily as complete or well organized as it should be, but is included to give a rough estimate for those interested.

Cost for device racks:

h914 rack with sockets			125.00
power bussing	1.25	8	10.00
unibus cable			115.00
power barrier strip			.61
wirewrapping			51.20

total			301.81
for 30 device cards:			603.62

Cost for distribution rack:

distribution rack:

rack			15.00
h803 sockets	14.00	5	70.00
power bussing	1.25	4	5.00
unibus cable to main unit			130.00
power barrier strip			.61
wirewrapping			48.20

rack subtotal			268.81
cost for row, col, clock, and driver cards			
may be guessed as:			250.00

total			518.81

Cost for speed select EIA interface card:

prices are for a quantity of 30 device cards

descr	each	#	total
16 pin sockets	.46	6	2.76
14 pin sockets	.40	10	4.00
1 40 pin socket	1.48	1	1.48

1 pc board			15.50
wirewrapping			16.50
wrap stakes			.10

subtotal			40.34

ics:			
uart			9.00
8t09			1.30
7408			.45
8t380	1.50	3	4.50
7474			.65
7403			.43
8t34	2.90	2	5.80
7400			.43
7410			.43
74151			1.43
mcl489			3.10
75150			2.80

subtotal			30.32

miscellaneous:			
1/4 watt 5% resistor (estimated)			.10
capacitor	.25	8	2.00
5 feet cord for tty			.28
jones plug on cord			.81
2 miniature cable ties at card			.04

subtotal			3.23

total			73.89
-------	--	--	-------

cost for 30 device cards:			2216.70
---------------------------	--	--	---------

Cost for main system unit:

system unit pricing:

h933cb frame and sockets			75.00
power bussing	1.25	4	5.00
wirewrapping			18.30

subtotal			98.30

data card:

pc board			37.50
16 pin sockets	.46	64	29.44
14 pin sockets	.40	21	8.40
wirewrapping			57.80
filter caps	.25	46	11.50
small caps	.15	2	.30
led (est)			1.00
12 volt zener (est)			1.50
2 resistors (est)			.20
delay lines (wild guess):			
25 nanoseconds			5.00
150 nanoseconds			5.00

subtotal			----- 157.64
----------	--	--	-----------------

ics:			
9112apc	3.70	11	40.70
74157	2.00	7	14.00
74s257	1.91	10	19.10
7475	1.15	2	2.30
74175	2.75	2	5.50
74193	3.30	9	29.70
74153	2.25	8	18.00
2841dc	9.50	2	19.00
8t38	3.63	9	32.67
8t34	2.90	2	5.80
8t09	1.30	2	2.60
7427	.80	3	2.40
8t380			1.50
7400	.43	4	1.72
7474	.65	2	1.30
7408	.45	3	1.35
7410			.43
7432			.80
7430	.55	2	1.10
7486	.85	1	.85
74h11			.90

subtotal			----- 201.72
----------	--	--	-----------------

total for data card			359.36
---------------------	--	--	--------

control card:

pc board			37.50
16 pin sockets	.46	9	4.14
14 pin sockets	.40	56	22.40
wirewrapping (est)			40.00
filter capacitors	.25	37	9.25
small caps	.15	6	.90

resistors (est)	.10	11	1.10
led (est)			1.00

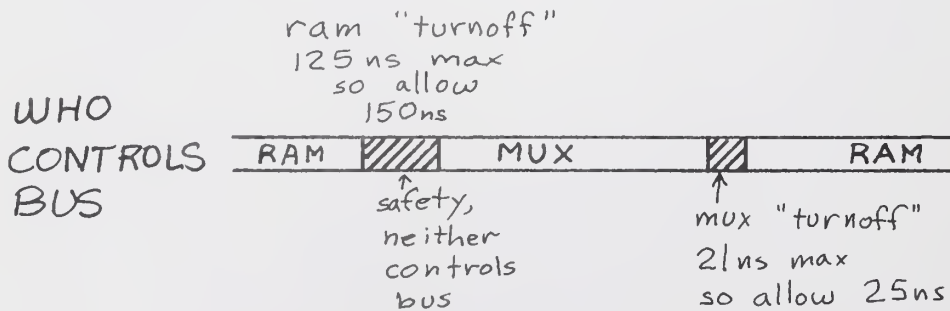
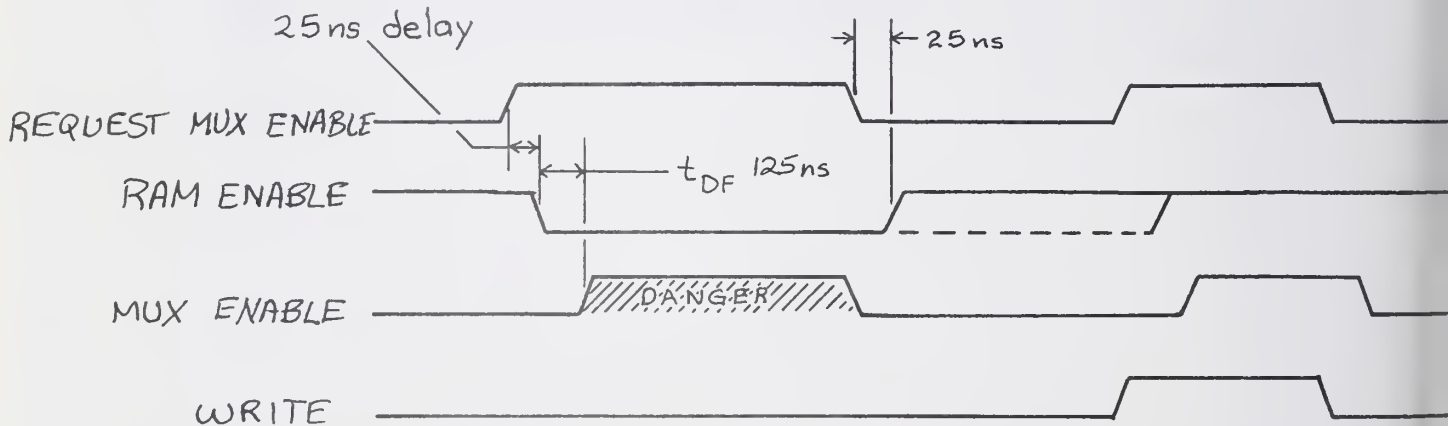
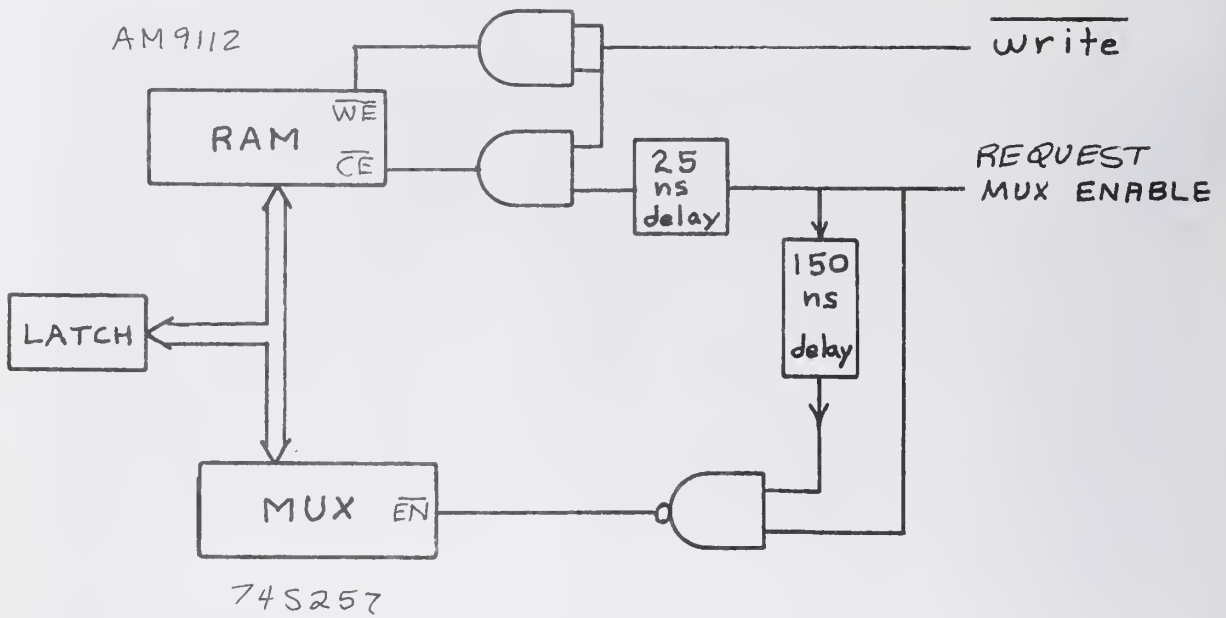
subtotal			116.29
ics:			
8t38	3.63	4	14.52
74175	2.75	4	11.00
74123	1.15	1	1.15
74h21			.90
7420	.55	2	1.10
7410	.43	2	.86
8t09	1.30	2	2.60
7400	.43	9	3.87
74h11	.90	1	.90
7474	.65	16	10.40
7408	.45	9	4.05
7437			1.15
7404	.70	4	2.80
7432			.80
7403	.43	2	.86
7425			.80
74107			.88
7412			.55
7409			.60
7430	.55	2	1.10

subtotal			60.89
total for control card			177.18
total for system unit			634.84

Appendix B -- failsafe system

The memory (RAMS) used in this multiplexor use a tristate bus on their four input/output lines to conserve pins and thereby package size. At the time of construction, 256x4 RAMS were very difficult to find and expensive but for these. In order to make proper use, a tristate multiplexor is used in conjunction with them as described elsewhere in this report. In order to provide maximum reliability, it was decided to have a special circuit to prevent conflicts and thereby avoid burning out chips or cause errors. The usual approach might be simply to attempt to guarantee absence of conflicts by making sure the much larger control circuitry never attempted to cause any. That approach would be more difficult since the chips in question have some special timing considerations. In addition the unit would be more prone to failure from component failure in the control circuitry, misplaced oscilloscope probe, or even from unplugging the control circuitry while the power was on, as the control is not on the same printed circuit card. The circuitry described here is all on the data card with the RAMS and the multiplexor chips.

The circuitry is fairly simple and straightforward as may be seen in the accompanying diagram which is largely self explanatory. The circuitry is certainly simpler than the manufacturers description of the timing considerations. The delays are introduced by delay lines which have the disadvantage of creating very sloppy waveforms but the advantage of being very reliable and of being bidirectional (zero to one as well as one to zero).

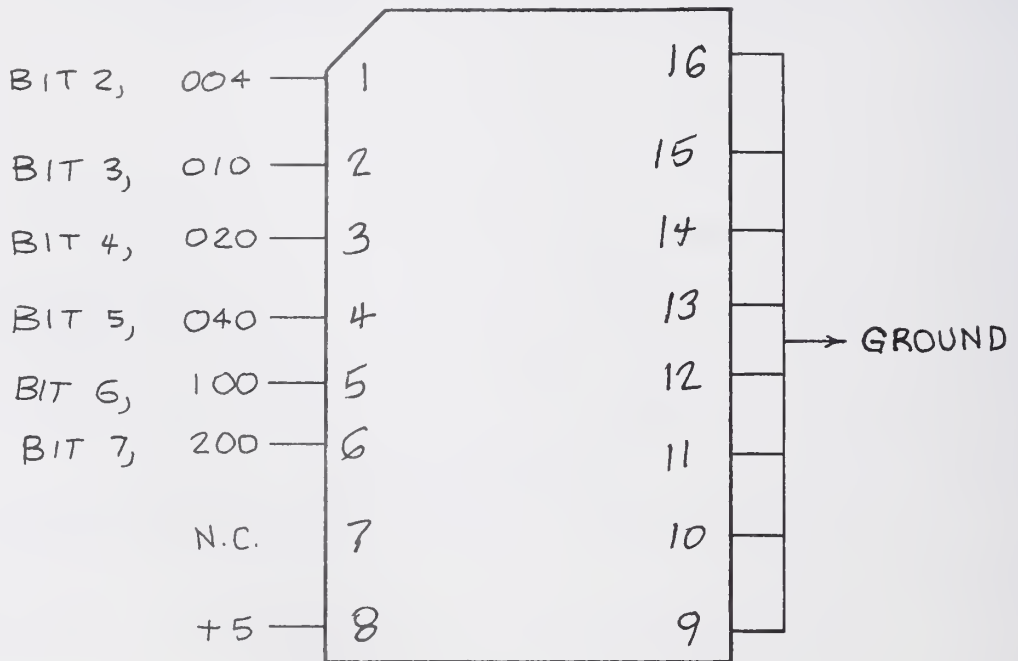


Appendix C -- address plugs

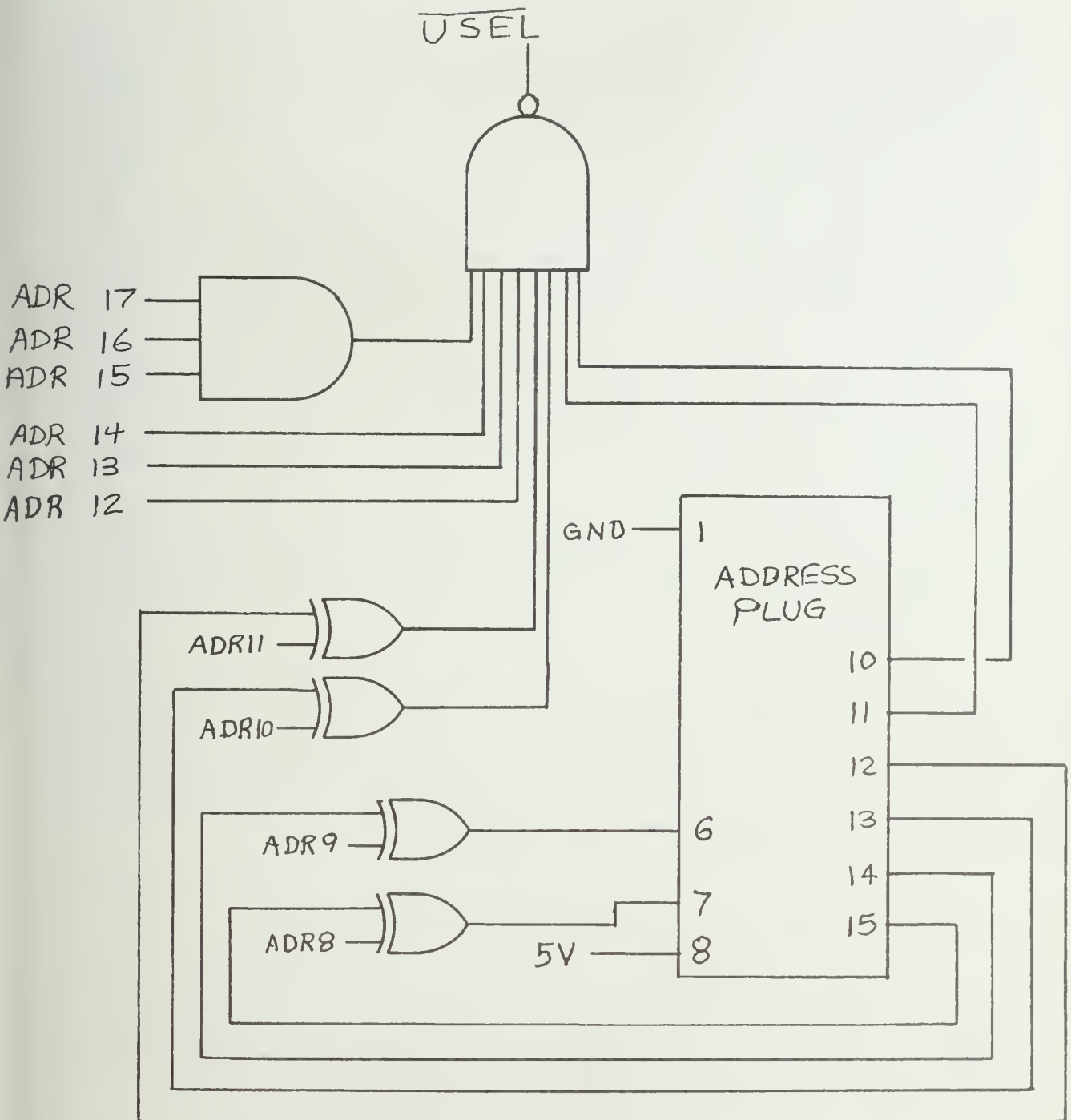
Two address plugs are provided on the data board to permit, as on most computer equipment of modern design, changing the address of the device and the address of the interrupt vector. The interrupt plug specifies where in the bottom 256 bytes of memory the new ps and pc are to be fetched from on interrupt. Since four bytes are required for the ps and the pc, the two bottom bits of the interrupt vector address must be zero, so they are not included in the address plug. Currently all multiplexor installations are using an address of 300 octal.

The Unibus address plug specifies field Y as discussed in chapter 5. The sixty four device addressing option may also be specified by appropriate connection on this plug. For 64 devices, connect pin 6 to pin 10 and pin 7 to pin 11. Bits 8 and 9 of the device address may then be specified. Ground on pin 14 would require bit 9 to be a one, five volts would require a zero. Pin 15 would specify bit 8 similarly. For 256 devices, pins 10 and 11 should be connected to five volts. Bits 10 and 11 are always be specified by pins 13 and 12 respectively.

INTERRUPT PLUG



MULTIPLEXOR UNIBUS ADDRESS PLUG



Appendix D -- EIA interface diagrams

Enclosed are diagrams for the speed select EIA interface discussed in chapter 15. The diagrams are provided to give a concrete example of how the circuit design looks for a typical interface. The interface is known internally by the name EIAD. Sublogic one is a circuit used in most of the interfaces built so far to communicate with the device bus. The telephone dialer and paper tape reader do not share sublogic one circuitry as they require only a subset of the circuitry. Sublogic two goes with sublogic one and is used to add a UART for serial communications. Sublogic three adds circuitry to sublogic two to allow hardware echo. This is known as half duplex in some circles, but as used in the UNIX software, it is enabled and disabled appropriately in order to provide full duplex communications at increased efficiency. The final diagram shows the speed select and EIA level shifting circuitry, as well as stop bit selection and cable plug wiring.

In sublogic one, requests for data transfers are made through the D1 chips which are open collector. Grants are received by B1 chips. The 4.7k resistor prevents devices from thinking they are selected if the device rack is unplugged from the distribution rack. Without the resistor, since the inputs float at zero, all the 8T34 chips would transmit with smoky results.

Requests obviously can not be made if the active bits in chips B2 and C2 are off, as this holds the request flipflops in C3 off. When device bus signal DSETREQ is issued, C3 is loaded with receiver request (RREQ) and transmitter request (TREQ) if the active bits are on. Transmitting and receiving are clearly totally independent.

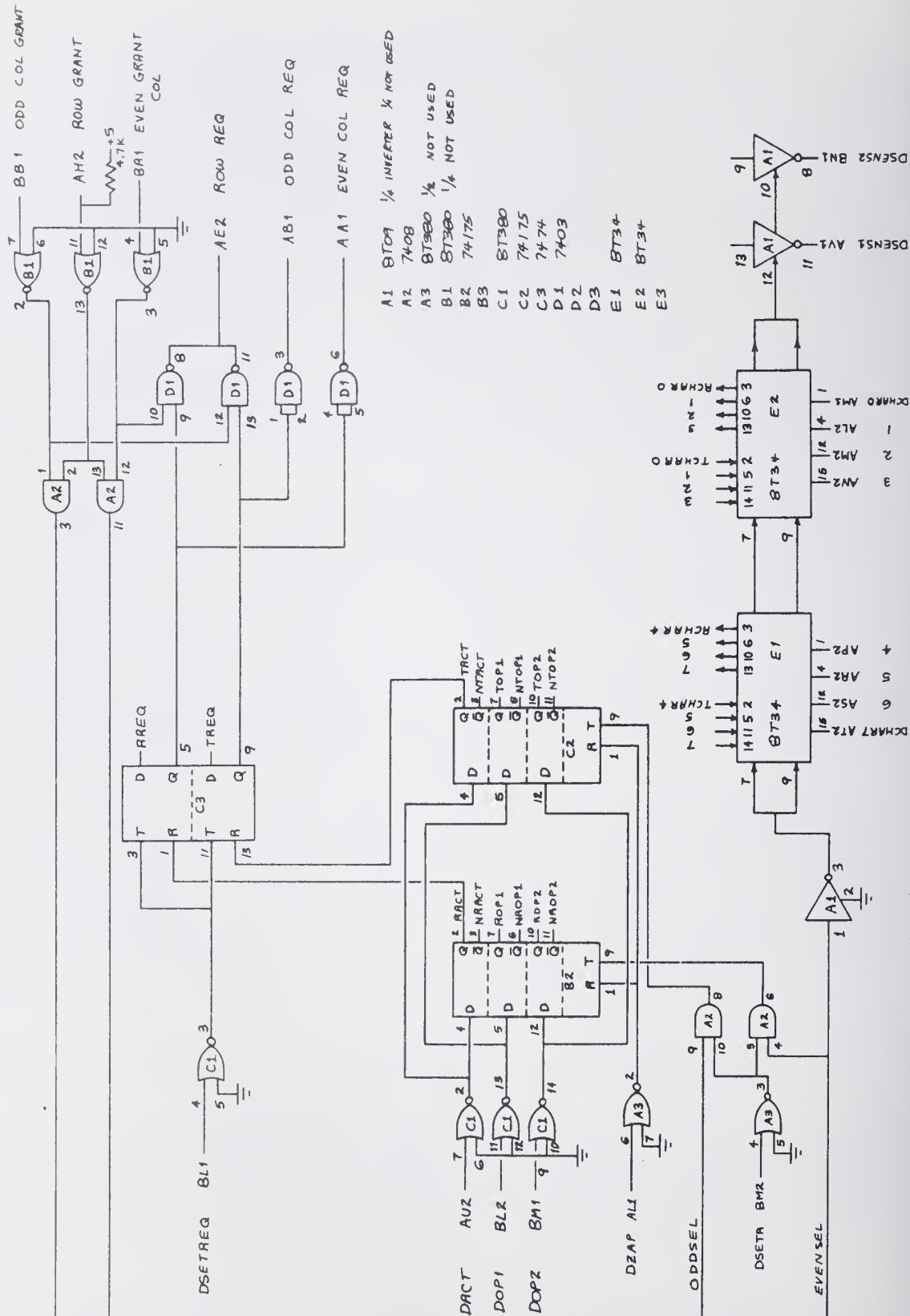
The signal DZAP causes the active bits and option bits to be cleared. No attempt was made to clear the UART in sublogic two as a matter of taste. The signal DSETA clearly loads the active bits and option bits on the selected device which may be one of the two devices in this interface or one outside of this interface. The sense outputs are not generally used, and the sublogic one circuit make provision for only two of the possible four. Two more could be provided with the same transmitters when the odd device was selected.

There are eight data lines entering (TCHAR) and eight leaving (RCHAR) this sublogic for use elsewhere in the interface.

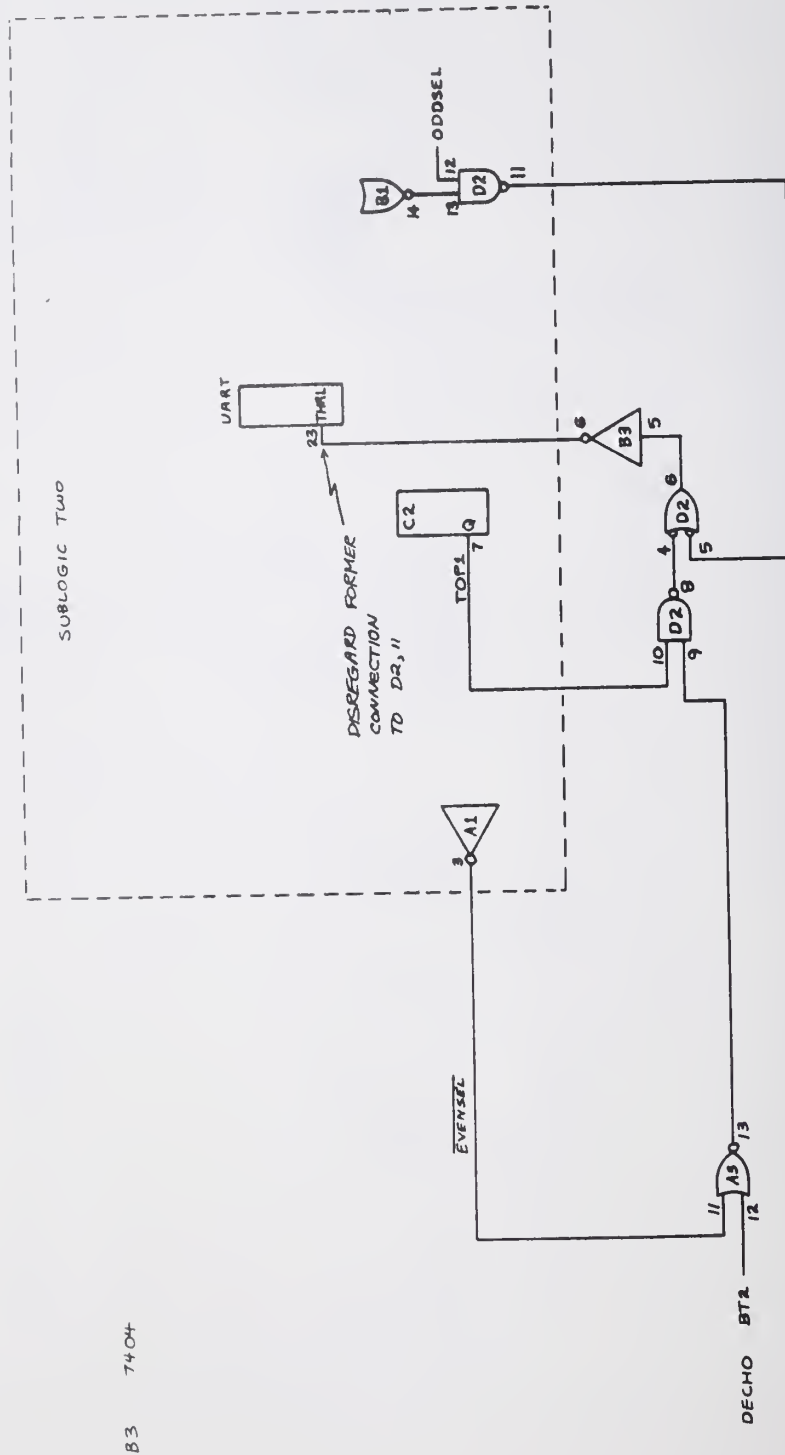
Sublogic two incorporates a UART rather straightforwardly. The device bus signal DRESET respectively clears the input flag for input or loads the output character for output. Connection of signals CLOCK, SERIAL IN, SERIAL OUT, are left for later diagrams. Other UART signals such as word length select, parity, etc. are hardwired to ground or five volts.

Sublogic three adds DECHO capabilities. This is an extra way to load the output side of the UART. The character transmitted is the received character as chips E1 and E2 are enabled.

The last drawing shows the speed select multiplexor, and the use of commercial EIA level shifters. One stop bit is selected for all but 110 baud in chip B3.



DEPARTMENT OF COMPUTER SCIENCE		TITLE		SUBLOGIC VERSION ONE	
University of Illinois		MULTIPLIER INTERFACE CARD		Sheet of Drawing No.	
Drawn by	WHALLEY	Date	7-18-78	Supervisor's sig.	
Approved by		Date			
Issue	Change and approved by	Date			



B3 7404

Appendix E -- Tables

Table 1. Multiplexor Bus From Mux to Distribution Rack Slot N

aal	devadr 1	bal	dopt 1
aa2		ba2	
abl	devadr 0	bb1	dopt 2
ab2	ground	bb2	ground
acl	devadr 2	bc1	dset a
ac2	ground	bc2	ground
ad1	devadr 3	bd1	ground
ad2	devadr 4	bd2	dreset
ael	devadr 5	be1	ground
ae2	devadr 6	be2	dact
afl	devadr 7	bf1	dsens 1
af2	adr 2 received	bf2	dsens 2
ahl	adr 3 received	bh1	zap
ah2	adr 4 received	bh2	load all latches (inverted)
aj1	adr 5 received	bj1	ck2
aj2	adr 6 received	bj2	dquit
ak1	adr 7 received	bk1	dterm
ak2	adr 8 received	bk2	
all	adr 9 received	b11	
al2	dcharbit 1	b12	
aml	dcharbit 0	bm1	
am2	dcharbit 2	bm2	
an1	ground	bn1	
an2	dcharbit 3	bn2	
apl	ground	bp1	
ap2	dcharbit 4	bp2	
arl	ground	br1	
ar2	dcharbit 5	br2	
asl	ground	bs1	
as2	dcharbit 6	bs2	
atl	ground	bt1	ground
at2	dcharbit 7	bt2	decho
aul	dinreq	bu1	
au2	doutreq	bu2	
avl	dsetreq	bv1	
av2	ground	bv2	

Table 2. Device Card Pin Assignment

aa1	column in 0 (even, read)	ba1	column out 0 (even, read)
aa2	+5 volts	ba2	+5 volts
ab1	column in 1 (odd, write)	bb1	column out 1 (odd, write)
ab2		bb2	
ac1		bc1	
ac2	ground	bc2	ground
ad1		bd1	
ad2		bd2	
ae1		be1	
ae2	row in	be2	clk 4800b
af1		bf1	clk 9600b
af2		bf2	clk 2400b
ah1		bh1	clk 1200
ah2	row out	bh2	clk 300
aj1		bj1	clk 110
aj2		bj2	clk 2000
ak1		bk1	
ak2	dquit	bk2	
al1	zap	bl1	d set req
al2	dcharbit 1	bl2	d opt 1
am1	dcharbit 0	bm1	d opt 2
am2	dcharbit 2	bm2	d set a
an1		bn1	d sens 2
an2	dcharbit 3	bn2	
ap1		bp1	
ap2	dcharbit 4	bp2	clk 38400b
ar1		br1	clk 19200b
ar2	dcharbit 5	br2	+12 volts
as1		bs1	
as2	dcharbit 6	bs2	-12 volts
at1	ground	bt1	ground
at2	dcharbit 7	bt2	decho
au1	dreset	bu1	dfinal
au2	dact	bu2	
av1	dsens 1	bv1	
av2		bv2	

Table 3. Device Numbers in Decimal
Given by their Position in the Priority Matrix

		COLUMN															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
	3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
	4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
	5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
R	6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
O	7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
W	8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
	9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
	10	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
	11	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
	12	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
	13	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
	14	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
	15	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Table 4. Device Numbers in Decimal
Given in Pseudo Priority Matrix Form
with their Rack Location

Note that the ninth rack has the missing corners of the
submatrices of the other racks.

rack 0:

0	1	2	3	4	5	6	7
16	17	18	19	20	21	22	23
32	33	34	35	36	37	38	39
48	49	50	51	52	53		

rack 1:

8	9	10	11	12	13	14	15
24	25	26	27	28	29	30	31
40	41	42	43	44	45	46	47
56	57	58	59	60	61		

rack 2:

64	65	66	67	68	69	70	71
80	81	82	83	84	85	86	87
96	97	98	99	100	101	102	103
112	113	114	115	116	117		

rack 3:

72	73	74	75	76	77	78	79
88	89	90	91	92	93	94	95
104	105	106	107	108	109	110	111
120	121	122	123	124	125		

rack 4:

128	129	130	131	132	133	134	135
144	145	146	147	148	149	150	151
160	161	162	163	164	165	166	167
176	177	178	179	180	181		

rack 5:

136	137	138	139	140	141	142	143
152	153	154	155	156	157	158	159
168	169	170	171	172	173	174	175
184	185	186	187	188	189		

rack 6:

192	193	194	195	196	197	198	199
208	209	210	211	212	213	214	215
224	225	226	227	228	229	230	231
240	241	242	243	244	245		

rack 7:

200	201	202	203	204	205	206	207
216	217	218	219	220	221	222	223
232	233	234	235	236	237	238	239
248	249	250	251	252	253		

rack 8:

54	55	62	63
118	119	126	127
182	183	190	191
246	247	254	255

Table 5. Unix Port Names and Device Numbers

suffix	port		devices			
	oct	dec	read		write	
			oct	dec	oct	dec
aa	0	0	0	0	1	1
ab	1	1	2	2	3	3
ac	2	2	4	4	5	5
ad	3	3	6	6	7	7
ae	4	4	10	8	11	9
af	5	5	12	10	13	11
ag	6	6	14	12	15	13
ah	7	7	16	14	17	15
ai	10	8	20	16	21	17
aj	11	9	22	18	23	19
ak	12	10	24	20	25	21
al	13	11	26	22	27	23
am	14	12	30	24	31	25
an	15	13	32	26	33	27
ao	16	14	34	28	35	29
ba	17	15	36	30	37	31
bb	20	16	40	32	41	33
bc	21	17	42	34	43	35
bd	22	18	44	36	45	37
be	23	19	46	38	47	39
bf	24	20	50	40	51	41
bg	25	21	52	42	53	43
bh	26	22	54	44	55	45
bi	27	23	56	46	57	47
bj	30	24	60	48	61	49
bk	31	25	62	50	63	51
bl	32	26	64	52	65	53
bm	33	27	66	54	67	55
bn	34	28	70	56	71	57
bo	35	29	72	58	73	59
ca	36	30	74	60	75	61
cb	37	31	76	62	77	63
cc	40	32	100	64	101	65
cd	41	33	102	66	103	67
ce	42	34	104	68	105	69
cf	43	35	106	70	107	71
cg	44	36	110	72	111	73
ch	45	37	112	74	113	75
ci	46	38	114	76	115	77
cj	47	39	116	78	117	79
ck	50	40	120	80	121	81
cl	51	41	122	82	123	83
cm	52	42	124	84	125	85
cn	53	43	126	86	127	87

suffix	port		devices			
	oct	dec	read		write	
			oct	dec	oct	dec
co	54	44	130	88	131	89
da	55	45	132	90	133	91
db	56	46	134	92	135	93
dc	57	47	136	94	137	95
dd	60	48	140	96	141	97
de	61	49	142	98	143	99
df	62	50	144	100	145	101
dg	63	51	146	102	147	103
dh	64	52	150	104	151	105
di	65	53	152	106	153	107
dj	66	54	154	108	155	109
dk	67	55	156	110	157	111
dl	70	56	160	112	161	113
dm	71	57	162	114	163	115
dn	72	58	164	116	165	117
do	73	59	166	118	167	119
ea	74	60	170	120	171	121
eb	75	61	172	122	173	123
ec	76	62	174	124	175	125
ed	77	63	176	126	177	127
ee	100	64	200	128	201	129
ef	101	65	202	130	203	131
eg	102	66	204	132	205	133
eh	103	67	206	134	207	135
ei	104	68	210	136	211	137
ej	105	69	212	138	213	139
ek	106	70	214	140	215	141
el	107	71	216	142	217	143
em	110	72	220	144	221	145
en	111	73	222	146	223	147
eo	112	74	224	148	225	149
fa	113	75	226	150	227	151
fb	114	76	230	152	231	153
fc	115	77	232	154	233	155
fd	116	78	234	156	235	157
fe	117	79	236	158	237	159
ff	120	80	240	160	241	161
fg	121	81	242	162	243	163
fh	122	82	244	164	245	165
fi	123	83	246	166	247	167
fj	124	84	250	168	251	169
fk	125	85	252	170	253	171
fl	126	86	254	172	255	173
fm	127	87	256	174	257	175
fn	130	88	260	176	261	177
fo	131	89	262	178	263	179

suffix	port		devices			
	oct	dec	read		write	
			oct	dec	oct	dec
ga	132	90	264	180	265	181
gb	133	91	266	182	267	183
gc	134	92	270	184	271	185
gd	135	93	272	186	273	187
ge	136	94	274	188	275	189
gf	137	95	276	190	277	191
gg	140	96	300	192	301	193
gh	141	97	302	194	303	195
gi	142	98	304	196	305	197
gj	143	99	306	198	307	199
gk	144	100	310	200	311	201
gl	145	101	312	202	313	203
gm	146	102	314	204	315	205
gn	147	103	316	206	317	207
go	150	104	320	208	321	209
ha	151	105	322	210	323	211
hb	152	106	324	212	325	213
hc	153	107	326	214	327	215
hd	154	108	330	216	331	217
he	155	109	332	218	333	219
hf	156	110	334	220	335	221
hg	157	111	336	222	337	223
hh	160	112	340	224	341	225
hi	161	113	342	226	343	227
hj	162	114	344	228	345	229
hk	163	115	346	230	347	231
hl	164	116	350	232	351	233
hm	165	117	352	234	353	235
hn	166	118	354	236	355	237
ho	167	119	356	238	357	239
ia	170	120	360	240	361	241
ib	171	121	362	242	363	243
ic	172	122	364	244	365	245
id	173	123	366	246	367	247
ie	174	124	370	248	371	249
if	175	125	372	250	373	251
ig	176	126	374	252	375	253
ih	177	127	376	254	377	255

Table 6. This is table 3 reproduced in octal

		COLUMN															
		0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
R O W	0	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
	1	20	21	22	23	24	25	26	27	30	31	32	33	34	35	36	37
	2	40	41	42	43	44	45	46	47	50	51	52	53	54	55	56	57
	3	60	61	62	63	64	65	66	67	70	71	72	73	74	75	76	77
	4	100	101	102	103	104	105	106	107	110	111	112	113	114	115	116	117
	5	120	121	122	123	124	125	126	127	130	131	132	133	134	135	136	137
	6	140	141	142	143	144	145	146	147	150	151	152	153	154	155	156	157
	7	160	161	162	163	164	165	166	167	170	171	172	173	174	175	176	177
	8	200	201	202	203	204	205	206	207	210	211	212	213	214	215	216	217
	9	220	221	222	223	224	225	226	227	230	231	232	233	234	235	236	237
	10	240	241	242	243	244	245	246	247	250	251	252	253	254	255	256	257
	11	260	261	262	263	264	265	266	267	270	271	272	273	274	275	276	277
	12	300	301	302	303	304	305	306	307	310	311	312	313	314	315	316	317
	13	320	321	322	323	324	325	326	327	330	331	332	333	334	335	336	337
	14	340	341	342	343	344	345	346	347	350	351	352	353	354	355	356	357
	15	360	361	362	363	364	365	366	367	370	371	372	373	374	375	376	377

Table 7. This is table 4 reproduced in octal

rack 0:

0	1	2	3	4	5	6	7
20	21	22	23	24	25	26	27
40	41	42	43	44	45	46	47
60	61	62	63	64	65		

rack 1:

10	11	12	13	14	15	16	17
30	31	32	33	34	35	36	37
50	51	52	53	54	55	56	57
70	71	72	73	74	75		

rack 2:

100	101	102	103	104	105	106	107
120	121	122	123	124	125	126	127
140	141	142	143	144	145	146	147
160	161	162	163	164	165		

rack 3:

110	111	112	113	114	115	116	117
130	131	132	133	134	135	136	137
150	151	152	153	154	155	156	157
170	171	172	173	174	175		

rack 4:

200	201	202	203	204	205	206	207
220	221	222	223	224	225	226	227
240	241	242	243	244	245	246	247
260	261	262	263	264	265		

rack 5:

210	211	212	213	214	215	216	217
230	231	232	233	234	235	236	237
250	251	252	253	254	255	256	257
270	271	272	273	274	275		

rack 6:

300	301	302	303	304	305	306	307
320	321	322	323	324	325	326	327
340	341	342	343	344	345	346	347
360	361	362	363	364	365		

rack 7:

310	311	312	313	314	315	316	317
330	331	332	333	334	335	336	337
350	351	352	353	354	355	356	357
370	371	372	373	374	375		

rack 8:

66	67
166	167
266	267
366	367

76	77
176	177
276	277
376	377

REFERENCE

- [1] "PDP11 Peripherals Handbook," Digital Equipment Corporation, Maynard, Massachusetts, 1975.

U. S. ATOMIC ENERGY COMMISSION
UNIVERSITY-TYPE CONTRACTOR'S RECOMMENDATION FOR
DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

1. AEC REPORT NO. COO-2383-0054	2. TITLE A PDP11 MULTIPLEXOR
--	-------------------------------------

3. TYPE OF DOCUMENT (Check one):

- ☒ a. Scientific and technical report
- ☐ b. Conference paper not to be published in a journal:
- Title of conference _____
- Date of conference _____
- Exact location of conference _____
- Sponsoring organization _____
- ☐ c. Other (Specify) _____

4. RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

- ☒ a. AEC's normal announcement and distribution procedures may be followed.
- ☐ b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.
- ☐ c. Make no announcement or distribution.

5. REASON FOR RECOMMENDED RESTRICTIONS:

6. SUBMITTED BY: NAME AND POSITION (Please print or type)

C. W. Gear
Professor and Principal Investigator

Organization

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

Signature



Date

September 1978

FOR AEC USE ONLY

7. AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

8. PATENT CLEARANCE:

- ☐ a. AEC patent clearance has been granted by responsible AEC patent group.
- ☐ b. Report has been sent to responsible AEC patent group for clearance.
- ☐ c. Patent clearance not required.

BIBLIOGRAPHIC DATA SHEET		1. Report No. UIUCDCS-R-78-942	2.	3. Recipient's Accession No.	
4. Title and Subtitle A PDP11 MULTIPLEXOR				5. Report Date September 1978	
				6.	
7. Author(s) A. D. Whaley				8. Performing Organization Rept. No. UIUCDCS-R-78-942	
9. Performing Organization Name and Address Department of Computer Science University of Illinois Urbana, Illinois 61801				10. Project/Task/Work Unit No.	
				11. Contract/Grant No. US ENERGY/EY-76-S-02-2383	
12. Sponsoring Organization Name and Address U.S. Department of Energy Chicago Operations Office 9800 South Cass Avenue Argonne, Illinois 60439				13. Type of Report & Period Covered research	
				14.	
15. Supplementary Notes					
16. Abstracts A multiplexor has been constructed for a DEC PDP11 intended for general purpose interfaces. Small interface boards are used for each connection to the multiplexor giving a heretofore unseen flexibility; virtually any byte oriented device may be easily connected. The interfaces are driven off a much simplified bus, keeping their circuit design simple. The single main unit performs all of the tasks of direct memory access, two character buffering, premature transfer stop conditions, multiplexing, maintaining separate device registers for each port, and handling automatic echo. A control point technique is employed, allowing the circuitry to resemble a wired program, making change easier. Modular construction has been used to allow the unit to be expanded in easy installments to a maximum of 128 full duplex ports.					
17. Key Words and Document Analysis. 17a. Descriptors multiplexor control point mini-computer					
17b. Identifiers/Open-Ended Terms					
17c. COSATI Field/Group					
18. Availability Statement unlimited				19. Security Class (This Report) UNCLASSIFIED	
				20. Security Class (This Page) UNCLASSIFIED	
				21. No. of Pages	
				22. Price	

DEC 7 1978



UNIVERSITY OF ILLINOIS-URBANA



3 0112 039311334